# CHAPTER - 1

# INTRODUCTION

## Overview of the Project

The Cold Storage Monitoring System using ESP32 is a cutting-edge implementation of IoT-based monitoring designed to address the critical need for reliable temperature and humidity control in environments where sensitive goods are stored. These environments include refrigerated warehouses, dairy product storage rooms, pharmaceutical chambers, and perishable food transport units. The core of this project revolves around the ESP32 microcontroller, a powerful and cost-effective microcontroller with integrated Wi-Fi capabilities. The ESP32 acts as the central brain of the system, acquiring real-time data from environmental sensors, processing this data, and making it available for both local and remote monitoring. A DHT11 or DHT22 sensor captures crucial environmental variables like temperature and humidity, which are vital for the preservation of perishable goods. These values are then displayed locally through an OLED screen or an I2C-enabled LCD module and simultaneously made accessible via a web interface hosted on the ESP32 itself. This allows users to track the condition of the cold storage room from any device connected to the same local network, ensuring timely intervention when necessary.

What distinguishes this system is its simplicity and adaptability. It does not depend on cloud platforms or mobile data connections, which may be unreliable or unavailable in remote areas. Instead, the ESP32 functions as a local webserver that can be accessed directly through an IP address assigned within the LAN. This makes the solution ideal for small-scale cold storages, especially in rural or semi-urban areas where affordability and offline capabilities are crucial. The system’s functionality can be further extended to include relays for turning on alarms, activating fans or heaters, or logging data for audit purposes. Overall, this project combines affordability, reliability, and real-time visibility, making it a strong candidate for deployment in small and medium-sized storage operations where temperature consistency.

## 1.2 Motivation

The driving force behind the development of the Cold Storage Monitoring System lies in the critical role that temperature-controlled environments play in modern-day food preservation, pharmaceutical safety, and agricultural post-harvest management. Across the globe—and particularly in developing countries like India—immense quantities of food are lost each year due to improper storage. For instance, India alone reports annual post-harvest losses exceeding ₹92,000 crore, mainly due to the unavailability of proper cold storage infrastructure and the lack of real-time monitoring. A similar situation exists in the pharmaceutical sector, where temperature-sensitive medicines, vaccines, and biologics must be stored within strict temperature ranges to maintain their potency and safety. A single breach in temperature threshold can render an entire batch of vaccines ineffective, endangering lives and incurring massive financial loss.

Despite these critical needs, most traditional cold storage systems are either manually monitored, leading to human error and delayed response, or rely on prohibitively expensive proprietary solutions that are out of reach for small-scale farmers, dairy cooperatives, or community medical centers. Moreover, with the advent of global warming and climate variability, even minor fluctuations in ambient conditions can impact cold room efficiency. Therefore, there is an urgent necessity to develop a monitoring system that is not only accurate and automated but also economical and accessible to a wider population. This project is rooted in the vision of democratizing smart technology—making it available to farmers, vendors, and administrators who lack access to sophisticated tools but still need precise monitoring and control over their cold storage units. The ESP32-based design was chosen specifically for its balance of performance and affordability, with the added advantage of Wi-Fi capabilities that allow seamless integration into existing infrastructure.

Our project aims to bridge the technological gap by providing a cost-effective, smart cold storage solution that's both efficient and accessible. By leveraging the ESP32's capabilities, we're democratizing smart technology for farmers, vendors, and administrators who previously lacked access to advanced tools. This innovative design combines performance, affordability, and Wi-Fi connectivity, seamlessly integrating into existing infrastructure

## 1.3 Problem Statement

Cold storage systems, while essential, often suffer from the fundamental problem of inadequate or delayed monitoring. In many cases, the temperature and humidity inside storage chambers are checked manually using analog thermometers or hygrometers, which offer no real-time visibility or recordkeeping. When values drift outside the safe range—whether due to equipment failure, power outages, or environmental fluctuations—the resulting spoilage is often detected too late. Moreover, these problems are exacerbated in remote or resource-limited settings, where staff may not be present around the clock, and expensive monitoring systems are simply not an option. Commercial solutions that provide real-time alerts and remote access typically come with high initial costs, complex installations, and recurring subscription fees for cloud connectivity, making them unfeasible for small cold rooms, village cooperatives, or agricultural produce markets.

In addition, these systems are heavily reliant on internet connectivity and mobile networks, which are not always dependable in rural or semi-urban locations. This introduces a critical need for a standalone, low-cost solution that can function effectively within a local area network (LAN) without any reliance on cloud servers. Furthermore, existing setups rarely offer user-friendly interfaces or the ability to interpret sensor data in meaningful ways. The absence of historical logging, notification triggers, and remote web dashboards makes them unsuitable for modern data-driven cold chain management. Therefore, the problem addressed by this project is the lack of an affordable, scalable, and easy-to-implement cold storage monitoring system that offers local and remote visibility, real-time alerts, and simple extensibility—all without needing continuous internet or a costly backend infrastructure.

A key advantage of IoT solutions is their capacity for immediate response. The system is designed to generate instant alerts and notifications, delivered via various channels including SMS text messages, email platforms, and dedicated mobile applications, whenever critical deviations or system errors are detected. This multi-channel alerting capability facilitates rapid intervention, allowing personnel to address issues promptly and mitigate potential losses. Furthermore, IoT systems automate the tedious processes of documentation and reporting.

## 1.4 Objectives

The primary goal of this project is to design, develop, and demonstrate a Cold Storage Monitoring System that leverages IoT technologies to provide real-time visibility into temperature and humidity conditions. This is to be achieved using cost-effective hardware components and open-source software tools. The system aims to maintain the viability of perishable items by ensuring that environmental parameters remain within acceptable limits and providing timely feedback if deviations occur. One of the central objectives is to allow both local and remote monitoring without the need for internet connectivity, relying instead on a simple LAN-based webserver hosted on the ESP32. The following are the detailed objectives of the project:

1. Develop a cost-effective cold storage monitoring system using ESP32, ensuring accessibility for widespread adoption and making it a viable solution for various industries.

2. Design the system for continuous operation, minimizing downtime and ensuring consistent data collection to provide accurate insights into the storage environment.

3. Utilize DHT11/DHT22 sensors to measure ambient temperature and humidity with reasonable accuracy, providing reliable data for monitoring and control purposes.

4. Display real-time sensor data on a local screen (OLED or LCD), enabling users to monitor conditions without relying on network connectivity and ensuring immediate awareness of any issues.

5. Develop a web interface accessible via Wi-Fi, allowing users to monitor real-time sensor data from any mobile or desktop device within the same network, providing flexibility and remote access.

6. Implement alert mechanisms (visual warnings or optional buzzers) that trigger when environmental parameters breach predefined limits, ensuring prompt action to prevent damage to stored goods.

7. Design a user-friendly and responsive web interface hosted on the ESP32, providing easy access to real-time data and system controls, and enabling users to navigate the system efficiently.

8. Ensure the system's modular design allows for future expansions, such as relay-based control for cooling units, fans, or alarm systems, making it adaptable to evolving needs.

## 1.5 Scope of the Project

The Cold Storage Monitoring System presented in this project is specifically targeted toward small-scale applications where affordability, simplicity, and effectiveness are paramount. It is not intended to replace industrial-grade systems but rather to provide a smart alternative where such infrastructure is lacking. The scope is limited to real-time monitoring of two key environmental parameters—temperature and humidity—which are critical to the maintenance of perishable goods. The system does not currently incorporate cloud storage, GSM modules for SMS alerts, or integration with third-party APIs, although the hardware and software architecture is designed to allow such features in future upgrades.

The user interface is confined to a local OLED or I2C LCD for real-time display and a Wi-Fi-based web interface for remote access within the LAN. The monitoring system assumes that the ESP32 has stable power supply and local network connectivity but does not require internet access. The relay-based actuator control, such as automatic activation of cooling units or exhaust fans, is not implemented in this basic version but can be easily added due to the modular design of the system. Furthermore, the system is best suited for indoor or semi-protected environments, as ruggedization against harsh weather or mechanical damage is beyond the current project's scope. Despite these constraints, the system achieves its core objective of enabling low-cost, efficient, and scalable cold storage monitoring, laying the foundation for future enhancements and broader applications in smart agriculture and logistics.

IoT systems transform documentation and reporting processes by automatically generating detailed logs and maintaining comprehensive digital audit trails, providing a transparent and tamper-proof record of all activities. This automation significantly streamlines regulatory compliance, making audits more efficient, less cumbersome, and less prone to errors, with readily accessible, organized, and accurate digital records. By minimizing manual effort and reducing the potential for human error, IoT enhances the accuracy, reliability, and consistency of compliance documentation, ensuring that organizations can maintain precise records and demonstrate adherence to regulatory requirements. As a result, organizations can mitigate risks associated with non-compliance, improve overall operational efficiency, and make informed decisions based on real-time data and insights.

# CHAPTER - 2

# LITERATURE REVIEW

## 2.1 Existing Cold Chain Solutions and Their Limitations

Cold chain systems are essential in a wide array of industries including agriculture, healthcare, food storage, logistics, and biotechnology. These systems ensure that products sensitive to temperature and humidity variations—such as vaccines, fruits, dairy, meats, and laboratory samples—are stored and transported under ideal environmental conditions. In developed countries, cold chains are managed through integrated infrastructure that includes refrigeration units, data loggers, cloud-connected monitoring devices, automated alert systems, and robust analytics platforms. However, the ground reality in many regions, especially in developing nations, is vastly different. The lack of investment, shortage of trained personnel, and limited access to reliable power and internet connectivity pose major challenges to the effective implementation of cold chain systems. In India, for example, the cold storage gap is enormous, and more than 30% of food produced post-harvest is lost due to improper handling and storage.

Commercial solutions from global vendors like Sensitech, Carrier Transicold, and Emerson offer sophisticated platforms with GPS tracking, GPRS-enabled alert notifications, and cloud-based data visualization. While these systems are technically sound and suitable for large-scale deployment, their high cost makes them inaccessible to small farmers, local food distributors, rural clinics, and decentralized vaccine centers. Furthermore, these systems often operate on a SaaS (Software as a Service) model with monthly subscriptions, making long-term affordability a concern. Another critical issue with these systems is the dependency on internet or mobile data, which can be unreliable or unavailable in remote areas. As a result, cold storage environments without internet access are often left unmonitored or depend solely on manual inspection and analog thermometers, which fail to provide real-time feedback or alerts in case of threshold violations. This gap creates a pressing need for decentralized, low-cost, and internet-independent cold storage monitoring systems—such as the one proposed in this project.

Another critical issue with these systems is the dependency on internet or mobile data, which can be unreliable or unavailable in remote areas.

## 2.2 Role of IoT in Cold Storage Monitoring

The Internet of Things (IoT) has rapidly emerged as a transformative technology across multiple domains, especially in real-time environmental monitoring and automation. In the context of cold storage management, IoT enables the remote sensing, control, and optimization of critical environmental parameters such as temperature, humidity, light exposure, and even air quality. By integrating microcontrollers like the ESP32 or Raspberry Pi with sensors and wireless communication modules, IoT systems can acquire live data from the field, transmit it over networks, and present it in an understandable format to users via web dashboards or mobile applications. The real-time nature of this monitoring ensures that deviations from desired thresholds can be identified and acted upon immediately, minimizing the risk of spoilage or damage.

IoT devices are particularly well-suited for cold storage systems because they are compact, power-efficient, and capable of local as well as remote data processing. Platforms like Blynk, ThingSpeak, and Firebase provide cloud integration for advanced solutions, but more lightweight and offline-capable systems like the ESP32-hosted webserver offer a practical alternative when network access is limited. Furthermore, by hosting local dashboards through HTTP servers, the need for expensive cloud infrastructure is eliminated. This not only reduces cost but also enhances data privacy and local autonomy. The ESP32, for instance, can be programmed to handle web requests, display sensor data, and even perform logic-based control (such as activating a fan when the temperature exceeds a threshold), all while operating entirely within a local network. Such features make IoT indispensable in democratizing access to real-time monitoring systems, particularly for small and mid-scale users who lack access to industrial-grade automation.

By hosting local dashboards via HTTP servers, costly cloud infrastructure is rendered unnecessary. This approach not only cuts expenses but also boosts data privacy and local control. The ESP32 exemplifies this capability, handling web requests, displaying sensor data, and executing logic-based controls—such as activating a fan when temperature thresholds are exceeded—all within a local network. This localized functionality empowers small to mid-scale users with limited access to industrial-grade automation, making real-time monitoring both accessible and affordable. Such advancements underscore IoT’s role in broadening technological access across diverse user groups.

## 2.3 Review of Sensor Technologies

At the heart of any environmental monitoring system lies the sensor technology responsible for collecting data. In cold storage applications, the key parameters are temperature and humidity, both of which directly affect the quality and shelf life of stored products. Over the years, numerous sensors have been developed to measure these parameters with varying degrees of accuracy, cost, and power consumption. Among them, the DHT11 and DHT22 sensors are particularly popular in low-cost embedded systems due to their simplicity, digital output, and ease of interfacing with microcontrollers. While DHT11 is affordable and sufficient for applications where high precision is not critical, the DHT22 offers improved accuracy and a wider range, making it more suitable for cold environments with extreme temperature fluctuations.

Beyond these, higher-end sensors like the SHT31, DS18B20, and BME280 provide even greater precision and include features such as dew point calculation, altitude estimation, and multi-sensor integration. However, they come at a higher cost and require more sophisticated calibration and software drivers. For the purposes of this project—targeting low-budget rural or semi-urban cold storage facilities—the DHT11 or DHT22 offers a sufficient trade-off between cost and performance. It provides temperature readings within ±2°C accuracy and humidity readings within ±5%, which are acceptable for most non-medical cold storage units. Furthermore, the digital nature of the sensor eliminates the need for analog-to-digital conversion, simplifying the microcontroller’s firmware and allowing it to focus on other tasks such as data serving and control logic

In contrast, high-end sensors come with a heftier price tag and demand intricate calibration and software expertise. For our project, targeting budget-friendly rural and semi-urban cold storage facilities, the DHT11/DHT22 strikes an ideal balance between affordability and performance. With temperature accuracy within ±2°C and humidity within ±5%, these sensors meet the requirements of most non-medical cold storage applications. Their digital output streamlines integration, eliminating the need for analog-to-digital conversion and allowing the microcontroller to prioritize critical tasks like data serving and control logic. This simplifies firmware development and enhances system reliability. Additionally, the plug-and-play nature of these sensors reduces setup time and complexity. Overall, the DHT11/DHT22 proves to be a practical and cost-effective solution for our specific use case, ensuring accessibility and efficiency.

## 2.4 Challenges in Existing Monitoring Systems

Despite the technological advancements in cold storage automation, several challenges remain in the field that limit the adoption of such systems at the grassroots level. The first major barrier is cost. Even the most basic commercial cold monitoring systems can cost thousands of rupees, which is unaffordable for local vendors, marginal farmers, and primary health centers. This high cost is often due to the inclusion of industrial-grade hardware, cloud service dependencies, and monthly maintenance contracts. The second challenge is connectivity dependency. Most systems rely heavily on internet availability, mobile networks, or cloud servers, making them unreliable in rural or isolated regions where such infrastructure is either unstable or completely absent.

Another issue is lack of user-friendliness. Many commercial monitoring dashboards are complex, filled with technical jargon and data analytics features that are unnecessary for users who simply want to know whether the cold room is functioning properly. These systems often require trained personnel for setup, interpretation, and maintenance. Additionally, maintenance and calibration of sensors is often overlooked in low-cost implementations, leading to false positives or undetected anomalies over time. Finally, the lack of interoperability between devices and the absence of open standards make it difficult to integrate components from different vendors or scale up the system over time. These limitations highlight the need for an open-source, locally accessible, modular, and user-friendly cold storage monitoring system that can be deployed with minimal training and resources.

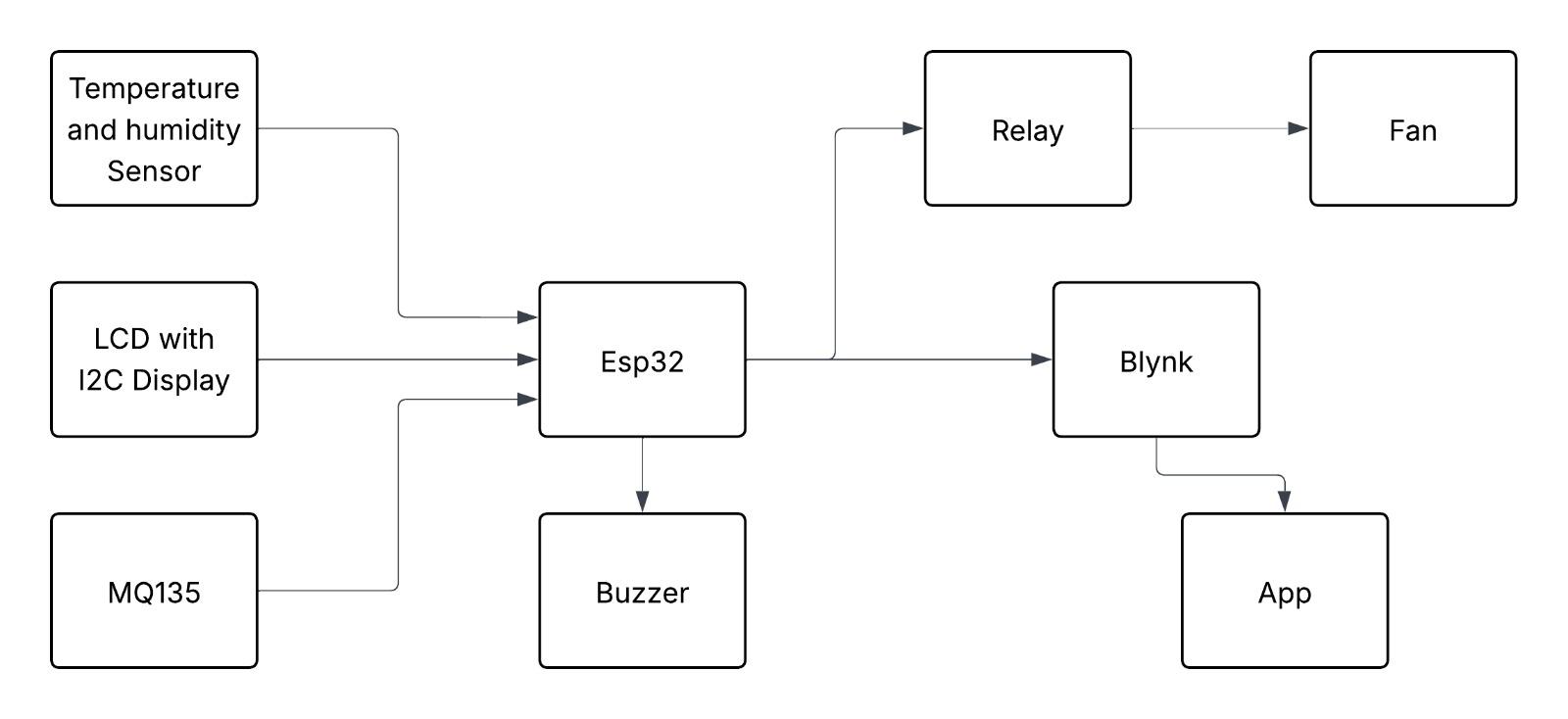
Traditional systems often rely on trained personnel for setup, operation, and maintenance, which can be a significant barrier for low-budget users. Neglecting sensor maintenance and calibration can lead to inaccurate readings, false alarms, or undetected issues. Furthermore, the lack of interoperability and open standards hinders integration with third-party devices and scalability. These challenges underscore the importance of developing an open-source, modular, and user-friendly cold storage monitoring system that is locally accessible, requires minimal training, and can be easily maintained and scaled. Such a system would empower users to efficiently monitor and manage their cold storage facilities without excessive reliance on technical expertise or vendor-specific solutions.

# CHAPTER - 3

**SYSTEM ARCHITECTURE AND DESIGN**

## 3.1 Block Diagram

The block diagram of the IoT-based cold storage monitoring system provides a high-level overview of the interaction between hardware components, sensors, and cloud-based applications used to monitor and control the storage environment. At the core of this system lies the ESP32 microcontroller, which serves as the central processing unit. It collects real-time environmental data from multiple sensors, processes it locally, and transmits the readings over Wi-Fi to the Blynk IoT platform, where users can remotely monitor temperature, gas levels, and lighting conditions via a smartphone app. This seamless integration of hardware and cloud services enables proactive monitoring and control without manual intervention.



## Figure 3.1 : Block Diagram

This sensor is selected for its compact design, digital output, and seamless integration with microcontroller platforms like the ESP32. The sensor data is transmitted to the ESP32, which processes and formats it for display on both local interfaces and remote platforms, ensuring real-time monitoring and accessibility.

The system incorporates several essential sensors to ensure the cold storage environment is kept within optimal thresholds. The **DHT11 sensor** is responsible for capturing temperature and humidity data, both of which are critical for preserving perishable goods. The **MQ135 gas sensor** monitors air quality by detecting harmful gases like ammonia or carbon dioxide, which may indicate spoilage or contamination. An **LDR (Light Dependent Resistor)** is used to monitor light levels, which can help detect if the storage room's door is opened or closed—vital for ensuring controlled access. Each sensor sends its respective signal to specific GPIO pins of the ESP32, enabling accurate and real-time data acquisition.

Based on the input data, the ESP32 takes automated actions. If the temperature rises above the predefined limit, the microcontroller activates a cooling fan through a relay module, thus maintaining the internal environment. In case of high gas concentration or abnormal conditions, an audible alert is triggered using a buzzer, notifying staff of potential risks. Simultaneously, real-time values are displayed on a 16x2 I2C LCD for local on-site monitoring. The LCD provides instant visibility into current system readings even without a smartphone.

Additionally, the ESP32 uses its built-in Wi-Fi capability to transmit data to the Blynk mobile app, allowing remote users to track live sensor values and receive alerts in case of emergency conditions. This dual-mode feedback system—local display and mobile interface—ensures both accessibility and redundancy. The block diagram reflects this architecture with clear interactions between sensors, the ESP32, output devices, and the cloud, highlighting the intelligent and responsive nature of the design. This architecture significantly improves efficiency and safety in cold storage operations, making it suitable for small-scale warehouses, food chains, and pharmaceutical storage.

## 3.2 System Architecture Overview

The architecture of the proposed cold storage monitoring system is designed around a modular, sensor-based framework that integrates IoT capabilities for real-time environmental control and remote monitoring. At the core is the ESP32 microcontroller, chosen for its dual-core processing, built-in Wi-Fi, and sufficient GPIOs to handle multiple sensor inputs and output devices. The system is divided into three main layers: the sensing layer, the processing/control layer, and the communication/interface layer. The sensing layer comprises sensors like DHT11 for temperature and humidity, MQ135 for gas detection, and an LDR for light intensity. These sensors continuously monitor the storage environment and provide analog or digital data to the ESP32, which then processes this data in real time to determine whether environmental parameters fall within predefined safe thresholds.

In the processing and control layer, the ESP32 not only analyzes sensor inputs but also makes decisions based on programmed logic. For example, if the temperature rises beyond a set point, the system triggers the relay module to switch on the cooling fan automatically. Similarly, if gas levels exceed safety limits or if abnormal light levels suggest unauthorized access, the buzzer is activated for an immediate alert. The final layer—the communication and interface layer—handles user interaction. Sensor readings are displayed locally using a 16x2 LCD for on-site operators, and are also transmitted to the Blynk mobile application over Wi-Fi. The Blynk platform enables remote monitoring, visualization, and alert management, making the system accessible from anywhere. This layered architectural approach not only ensures modularity and scalability but also supports real-time automation, enhancing the reliability and safety of cold storage management.

## 3.3 Flow of Operation

The operation of the cold storage monitoring system begins with the initialization of the ESP32 microcontroller, which establishes a Wi-Fi connection and prepares the GPIO pins to interface with the connected sensors and output devices. Once the system powers on, it starts collecting real-time data from the environmental sensors. The DHT11 sensor measures the current temperature and humidity, the MQ135 gas sensor detects the concentration of harmful gases, and the LDR senses the ambient light level within the cold storage room. These values are read through dedicated analog or digital pins on the ESP32 and stored temporarily for decision-making.

After the data is collected, the ESP32 processes it through conditional logic defined in the firmware. If the temperature crosses a preset threshold, the controller activates a relay connected to a fan to cool the storage environment. If the MQ135 detects a gas concentration beyond safe levels, it triggers a buzzer to alert personnel immediately.

The LDR data is used to detect whether there is unexpected light inside the cold storage, which could indicate that the door is open or someone has accessed the area. These automated responses ensure that conditions are promptly corrected and any deviation from safety parameters is quickly addressed.

Simultaneously, the system sends the collected sensor data to the Blynk mobile application through the internet. This allows users to remotely monitor the temperature, humidity, gas levels, and light intensity on their smartphones in real time. Blynk also provides alert notifications if the system detects abnormal conditions. Additionally, a local 16x2 LCD display connected to the ESP32 shows the same readings on-site, offering a quick visual reference for workers in the storage area. This seamless combination of data acquisition, decision-making, automated control, and dual-mode data display creates a reliable, responsive, and user-friendly flow of operation that enhances both safety and efficiency in cold storage management.

## 3.4 Circuit Diagram

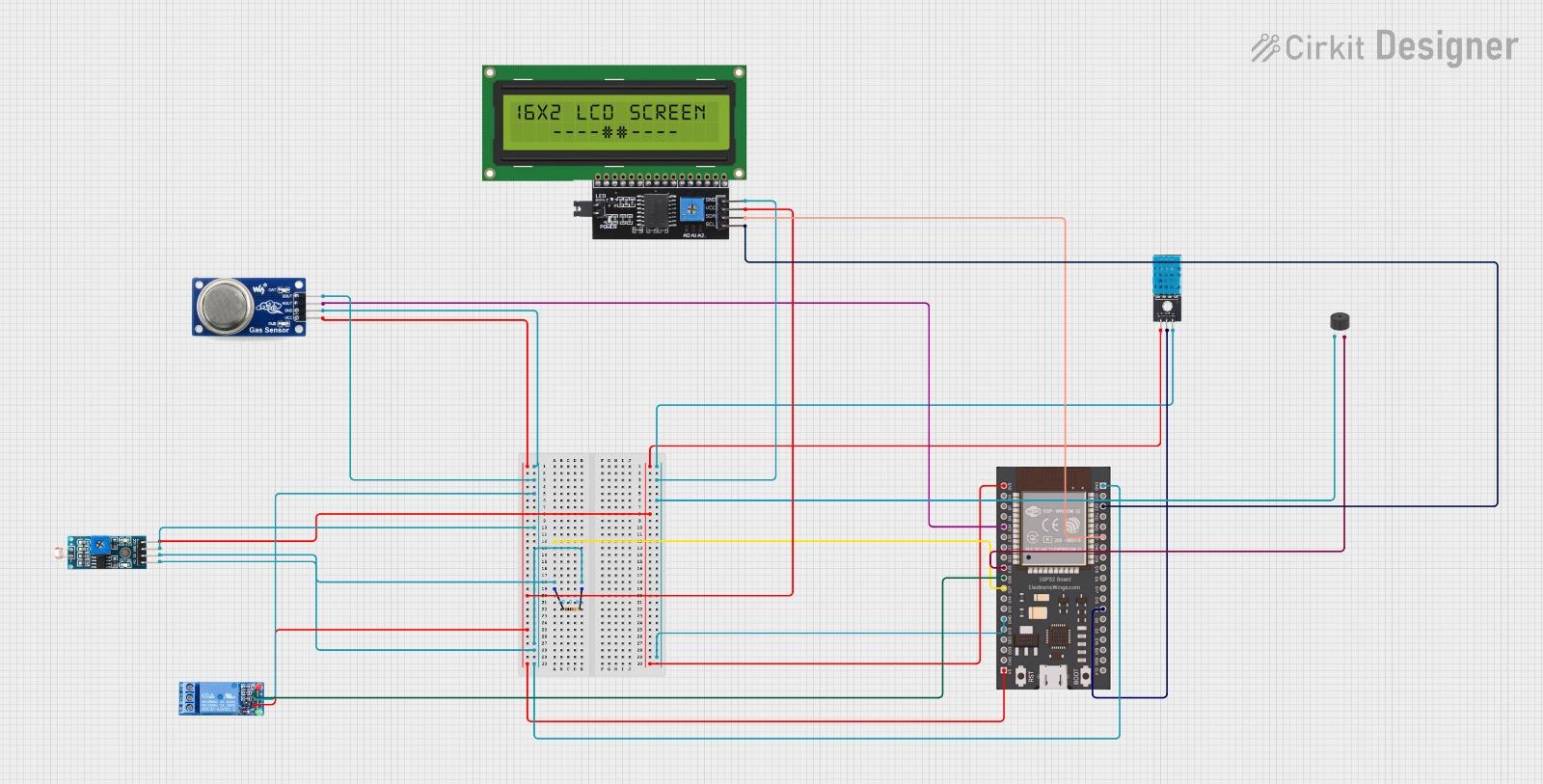


Fig 3.4 Circuit Diagram

The hardware design of the Cold Storage Monitoring System is intentionally kept simple and modular to facilitate easy prototyping and future transition to a printed circuit board (PCB).

At the heart of the system is the ESP32 Development Board, which operates on 3.3V logic and is powered via the Vin pin from a 5V source. If a 12V DC adapter is used as the power supply, a buck converter is incorporated to step down the voltage to 5V. The onboard LDO regulator of the ESP32 further reduces this to 3.3V to power internal components and 3.3V-compatible sensors like the DHT11. The DHT11 sensor is connected to a digital GPIO pin (e.g., GPIO 4) and uses a 10kΩ pull-up resistor between its data line and VCC to ensure stable digital signals.

A 16x2 LCD with I2C interface is used to display real-time data such as temperature, humidity, and gas concentration. The I2C interface minimizes pin usage and wiring complexity by utilizing only two GPIOs—GPIO 21 (SDA) and GPIO 22 (SCL)—to communicate with the ESP32. These connections are standardized across many ESP32 development boards. Power (VCC and GND) is distributed using a breadboard power rail, allowing multiple sensors and output modules to share common lines. This approach not only ensures efficient use of space but also reduces potential errors during assembly.

The system includes a Relay Module connected to the ESP32 via a digital pin (e.g., GPIO 26) and is used to drive a cooling fan. The relay circuit is powered by the 5V regulated supply and is often triggered using a BC547 transistor to provide sufficient current to activate the relay coil. To protect the circuit from voltage spikes during switching, a flyback diode (such as 1N4007) is placed across the relay coil. This ensures long-term reliability and prevents back EMF from damaging the microcontroller. Additionally, a buzzer is included as an audible alert mechanism, activated when gas levels or temperature exceed defined safety thresholds. This buzzer is connected to GPIO 25, drawing power from the 5V rail and grounded to the system ground.

The MQ135 gas sensor is connected to the analog input GPIO 34 on the ESP32 and powered via the 5V rail. This sensor plays a key role in detecting air quality and early signs of food spoilage or contamination due to gases like ammonia or CO2. An LDR (Light Dependent Resistor) is integrated into the system to detect light levels and infer door activity. It is connected using a voltage divider configuration with a 10kΩ resistor, and the junction is fed into GPIO 27, allowing the ESP32 to detect sudden changes in light that may indicate unauthorized access or temperature loss due to an open door.

The overall component layout on the breadboard is designed to minimize wire length, reduce EMI, and maintain organized spacing for thermal safety. For industrial deployment, this circuit can be migrated to a custom PCB design with screw terminals, shielding, and fuses to enhance reliability. The PCB version can incorporate features like overcurrentprotection, voltage filtering, and standardized connectors, making it suitable for long-term use in demanding storage environments. Such a design ensures modularity, easy maintenance, and future scalability, which is ideal for distributed storage systems or remote warehouse monitoring.

The MQ135 gas sensor, connected to the ESP32's GPIO 34, plays a crucial role in monitoring air quality within the cold storage facility. Powered by the 5V rail, it detects gases like ammonia and CO2, which can be early indicators of food spoilage or contamination. This real-time monitoring capability enables prompt action to prevent losses and ensure product safety.

In parallel, a Light Dependent Resistor (LDR) is integrated into the system to detect changes in light levels, which can signify door activity or potential security breaches. Configured in a voltage divider setup with a 10kΩ resistor, the LDR's output is fed into GPIO 27 on the ESP32. This setup allows the system to infer if the door is left open or if there's unauthorized access, triggering appropriate alerts or responses to maintain the integrity and security of the storage environment. By combining gas detection and light monitoring, the system provides comprehensive oversight of the cold storage conditions.

The system incorporates a Relay Module, connected to the ESP32 via a digital pin like GPIO 26, to control a cooling fan. The relay circuit operates on the 5V regulated supply and is triggered using a transistor like the BC547, ensuring sufficient current to activate the relay coil. A flyback diode (1N4007) across the relay coil protects the circuit from voltage spikes during switching, preventing back EMF from damaging the microcontroller and ensuring long-term reliability.

In addition to temperature control, the system features an audible alert mechanism—a buzzer connected to GPIO 25. Powered by the 5V rail and grounded to the system, the buzzer is activated when gas levels or temperatures exceed predefined safety thresholds. This auditory warning system ensures that personnel are promptly notified of potential issues, allowing for timely intervention to maintain safe storage conditions. By integrating relay-controlled devices and alert mechanisms, the system provides a robust solution for managing cold storage environments, enhancing both product safety and operational efficiency.

## 3.5 Circuit Description

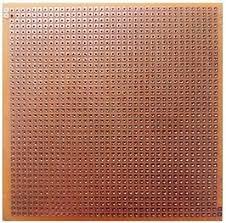
System has been carefully designed to ensure simplicity, reliability, and compatibility with low-cost components. At the heart of the system lies the ESP32 development board, which serves as the central processing and communication unit. It operates at 3.3V logic but can be powered through its Vin pin using a 5V regulated supply, making it compatible with common USB adapters and mobile chargers. The DHT11 or DHT22 temperature and humidity sensor is connected to a digital GPIO pin (commonly GPIO 15), and a 10kΩ pull-up resistor is placed between the sensor’s data line and the 3.3V rail to stabilize the digital signal. The sensor requires three connections: VCC (3.3V), GND, and DATA. For real-time visual feedback, a 16x2 LCD or a 0.96” OLED display is connected using the I2C protocol, which uses GPIO 21 (SDA) and GPIO 22 (SCL) on the ESP32. These I2C lines are shared between compatible devices, simplifying the wiring and reducing pin usage.

To enhance user interaction and enable visual alerts, the system may include an optional buzzer or LED connected to a separate GPIO pin. For relay control, a 5V single-channel relay module can be added to manage external devices like fans or alarms. This relay is triggered by a digital output from the ESP32 (e.g., GPIO 26), and a transistor (such as 2N2222 or BC547) may be used to drive the relay if current amplification is needed. A flyback diode like 1N4007 is placed across the relay coil to protect the circuit from voltage spikes caused by inductive loads. The entire system is powered via a 5V adapter, and if a 12V adapter is used, a buck converter or LM7805 voltage regulator is employed to step the voltage down safely. All connections are initially made on a breadboard for prototyping, and in final deployment, a custom PCB is recommended to secure the components, ensure electrical safety, and enhance durability. The design ensures stable operation and provides flexibility for expansion, making it ideal for continuous cold storage monitoring in practical environments. The ESP32’s onboard capabilities allow it to function as a complete HTTP web server, enabling users to access environmental data in real-time from any device connected to the same local Wi-Fi network. Upon boot-up and successful Wi-Fi connection, the ESP32 hosts a web page that displays the current temperature and humidity values, along with optional indicators or alerts if these values exceed predefined safety limits. The web interface is written in basic HTML and CSS and embedded directly into the ESP32’s flash memory using PROGMEM directives. This embedded design ensures lightweight execution and fast response times without the need for external hosting or reliance on internet-based platforms.

The server refreshes data periodically, providing live feedback and allowing users to monitor cold room conditions conveniently from smartphones, tablets, or computers. This feature enhances system accessibility and makes remote supervision practical in a LAN environment.

## 3.6 PCB/Breadboard Layout

The physical arrangement of components, whether on a breadboard for prototyping or a Printed Circuit Board (PCB) for a permanent solution



**Breadboard Layout (Prototyping Phase):** During the initial development and testing phases, a breadboard provides a flexible and solder-less environment for assembling the circuit.

* + 1. **Ease of Modification:** Components can be easily inserted, removed, and reconfigured, which is ideal for experimentation and debugging.
    2. **Visibility:** The open nature of a breadboard allows for clear visualization of connections, aiding in troubleshooting.
    3. **Wiring:** Jumper wires are used to connect components. It is crucial to use short, secure jumper wires to minimize noise interference, especially for sensor readings.
    4. **Power Rails:** The breadboard's power rails (typically along the sides) are used to distribute VCC (e.g., 5V and 3.3V from voltage regulators) and GND connections to all components.
    5. **Component Placement:** Components are placed logically to minimize wire lengths and avoid signal crossovers. For instance, the DHT11 sensor should be placed where it can accurately measure ambient conditions, and the LCD should be easily visible. The relay module, especially if handling higher currents, should be somewhat isolated from sensitive digital components to prevent electromagnetic interference.

**PCB Layout (Final Product/Permanent Solution):** For a robust, compact, and reliable final product, transitioning from a breadboard to a custom PCB is highly recommended.

1. **Compactness:** PCBs allow for highly compact designs, integrating all components onto a single board, which is essential for embedded systems.
2. **Reliability:** Soldered connections on a PCB are far more reliable and less prone to intermittent issues (e.g., loose wires) than breadboard connections.
3. **Signal Integrity:** Traces on a PCB can be precisely designed to minimize noise, impedance mismatches, and signal degradation, which is critical for high-frequency signals (like Wi-Fi) and sensitive analog readings.
4. **Power Distribution:** PCB traces can be made wider for power lines (VCC and GND) to handle higher currents and minimize voltage drops across the board.
5. **Thermal Management:** Components that generate heat, such as voltage regulators (e.g., 7805) and power transistors (e.g., TIP122 for the pump, if used), require adequate copper pour or dedicated heatsinks on the PCB. Proper thermal design prevents overheating and ensures component longevity.
6. **Ground Planes:** Implementing a solid ground plane on a multi- layer PCB significantly improves noise immunity and signal integrity by providing a low-impedance return path for currents.
7. **Component Placement for Functionality:** For the Smart Plant Monitoring System, the PCB layout would consider:
   * Placing the ESP32 centrally or in a location that optimizes trace lengths to other components.
   * Keeping the DHT11 sensor away from heat-generating components (like voltage regulators or the relay) to ensure accurate temperature readings.
   * Positioning the LCD for easy viewing.
   * Ensuring adequate clearance and isolation for the relay module, especially if it switches AC loads, to prevent electrical hazards and interference.
   * Providing mounting holes for securing the PCB within an enclosure.

The article mentions the availability of top and bottom PCB layouts and Gerber files for ordering custom PCBs, or instructions for creating a

homemade PCB on a zero PCB, indicating that detailed physical design considerations are part of the project's practical implementation. This attention to physical layout is crucial for transforming a functional prototype into a dependable and deployable system.

## 3.7 Power Supply Management

Effective power supply management is paramount for the stable and reliable operation of the Smart Plant Monitoring System, particularly when integrating components with diverse voltage requirements. The system is designed to operate from a 12V DC power adapter, which necessitates careful voltage regulation to meet the specific needs of the ESP32 microcontroller and other peripherals.

The primary power input to the system is a 12V DC, 1A (or higher, depending on the pump's current rating) power adapter. This voltage is suitable for driving the relay module directly if it operates at 12V, or for powering a 12V water pump. However, the ESP32 microcontroller operates at a 3.3V logic level, and its development boards typically require a 5V input on their Vin pin. Directly applying 12V to the ESP32's Vin pin can cause significant overheating and irreversible damage, as the onboard voltage regulators are generally linear and designed for smaller voltage drops (e.g., 5V to 3.3V).

To address this, a **step-down voltage regulation stage** is implemented:

### 12V to 5V Conversion:

* + - * **Buck Converter (Recommended):** The most efficient and recommended method for stepping down 12V to 5V is using a buck converter (also known as a switching regulator). These modules are highly efficient (typically 80-95%), converting excess voltage by rapidly switching the input on and off, which minimizes energy loss as heat. They can handle higher current loads (e.g., 1A to 3A) and are ideal for power-sensitive applications. The 12V input is connected to the buck converter's IN+ and IN- terminals,

and its OUT+ (5V) and OUT- (GND) are then connected to the ESP32's Vin and GND pins, respectively. If using an adjustable buck converter, its output voltage must be precisely set to 5V using a multimeter and a potentiometer.

* + - * **LM7805 Linear Regulator (Alternative):** A linear regulator like the LM7805 can also convert 12V to 5V. However, linear regulators dissipate the excess voltage as heat, making them less efficient (40-60%) and prone to overheating, especially with larger voltage differences or higher current draws. If a 7805 is used, it is absolutely critical to employ an adequate **heatsink** to manage the dissipated heat and prevent thermal shutdown or damage to the regulator and surrounding components. The 5V output from the 7805 then feeds the ESP32's Vin pin.

### 5V to 3.3V Conversion:

* + - * Most ESP32 development boards come with an **onboard**

**3.3V LDO (Low-Dropout) voltage regulator**. This regulator takes the 5V supplied to the Vin pin (whether from USB or an external 5V source like a buck converter/7805) and steps it down to the precise 3.3V required by the ESP32's microcontroller core and other 3.3V-compatible components like the DHT11 sensor (if its VCC is connected to 3.3V).

**Current Capacity Considerations:** The current rating of the 12V power adapter and the chosen voltage regulators must be sufficient to power all components. While the ESP32 typically consumes 100-300mA (with peaks during Wi-Fi transmission), the total current draw increases significantly when driving peripherals like the LCD, sensors, and especially the water pump via the relay. A buck converter capable of supplying at least 1A is generally sufficient for basic projects, but for setups with multiple peripherals or a powerful pump, a 2A or 3A converter is recommended to ensure stable performance and prevent.

voltage drops. The amperage rating of the 12V power supply should specifically match the current requirements of the water pump or solenoid valve.

The current rating of the 12V power adapter and voltage regulators is a critical consideration to ensure all components operate reliably. The ESP32, a central component in this system, typically consumes between 100-300mA, with current spikes during Wi-Fi transmission or when handling intensive tasks. However, when peripherals such as LCDs, various sensors, and particularly relay-driven devices like cooling fans or water pumps are added to the system, the total current draw increases significantly.

To manage this increased load effectively, it's essential to select a buck converter that can supply sufficient current to meet the demands of all connected devices. For basic projects with minimal peripherals, a buck converter rated for at least 1A might be adequate. However, for more complex setups involving multiple sensors, displays, and high-power devices, opting for a converter with a higher current rating—such as 2A or 3A—becomes necessary. This ensures that the system remains stable under varying load conditions and prevents potential issues like voltage drops, overheating, or erratic behavior.

In industrial or remote applications where system reliability is paramount, proper power management is especially critical. System failures or downtime in such environments can lead to significant financial losses, safety hazards, or operational disruptions. By carefully selecting power components with adequate ratings and designing the system for worst-case scenarios, engineers can build robust and reliable systems that meet both performance and safety requirements, ensuring long-term functionality and minimizing the need for maintenance or repairs.

By choosing components with adequate ratings, designers can build robust systems that meet performance and safety requirements.

# CHAPTER - 4

**HARDWARE & SOFTWARE COMPONENTS DESCRIPTION**

## ESP32 Microcontroller



Fig 4.1 ESP32 Microcontroller

The ESP32 is a highly integrated, low-cost, and energy-efficient System- on-Chip (SoC) designed by Espress if Systems and manufactured by TSMC using their 40 nm process. It serves as the successor to the popular ESP8266 and is widely adopted in IoT applications due to its robust features and dual connectivity options. For the Smart Plant Monitoring System, the ESP32 acts as the central processing and communication unit, orchestrating all data flow and control logic.

### Key Specifications and Features:

* + 1. **Processors:** The ESP32 typically features a powerful Xtensa dual-core (or single-core) 32-bit LX6 microprocessor. These cores can operate at clock frequencies of 160 MHz or 240 MHz, delivering up to 600 DMIPS (Dhrystone Million Instructions Per Second) of processing power. Additionally, it includes an ultra-

low-power (ULP) co-processor, which can handle basic tasks in deep sleep mode, contributing to energy efficiency. Specific variants like the ESP32-S3 utilize a dual-core Xtensa LX7 CPU with single-precision FPU and added instructions for machine learning acceleration, showcasing the family's versatility.

* + 1. **Memory:** The standard ESP32 comes with 520 KiB of SRAM (Static Random-Access Memory) and 448 KiB of ROM (Read-Only Memory). Variants like the ESP32-S3 offer 512 KiB SRAM and 384 KiB ROM, with additional RTC SRAM. The ability to connect to external PSRAM and Flash via SPI interfaces (Quad SPI or Octal SPI) significantly expands its memory capabilities, allowing for larger programs and data storage.
    2. **Wireless Connectivity:** A standout feature of the ESP32 is its integrated wireless capabilities. It supports Wi-Fi (802.11 b/g/n) for robust network connectivity and Bluetooth (v4.2 BR/EDR and BLE - Bluetooth Low Energy), sharing the same radio. Newer variants like the ESP32-S3 support Bluetooth 5 (LE) and even IEEE 802.11ax (Wi-Fi 6) in the ESP32-C6, offering enhanced performance and lower power consumption. This dual connectivity makes it ideal for IoT applications requiring both local and network communication.
    3. **Peripheral Interfaces:** The ESP32 boasts a rich set of peripheral interfaces, providing extensive connectivity options for various sensors and actuators. It typically includes 34 programmable General-Purpose Input/Output (GPIO) pins, 10 capacitive touch sensors, 2 × 12-bit SAR ADCs (up to 18 channels), and 2 × 8-bit DACs (except on some variants). Standard communication interfaces include 4 × SPI, 2 × I²S, 2 × I²C, and 3 × UART. Other features like Motor PWM, LED PWM (up to 16 channels), and a CAN bus 2.0 further enhance its utility in diverse embedded projects.
    4. **Security Features:** Security is a critical aspect of IoT devices, and the ESP32 incorporates robust features. It supports IEEE

802.11 standard security protocols, including WPA, WPA2, and

WPA3 (depending on the version). Hardware-accelerated cryptography (AES, SHA-2, RSA, ECC, RNG), secure boot, and flash encryption provide a strong foundation for protecting data and firmware.

* + 1. **Power Management:** The ESP32 is designed for low-power applications, featuring an internal low-dropout regulator and individual power domains for its Real-Time Clock (RTC). It can achieve a deep sleep current as low as 5 μA and can be woken up from various interrupts, including GPIO, timers, ADC measurements, and capacitive touch sensors. This power efficiency is vital for battery-powered or energy-conscious IoT deployments.

## DHT11 Temperature & Humidity Sensor

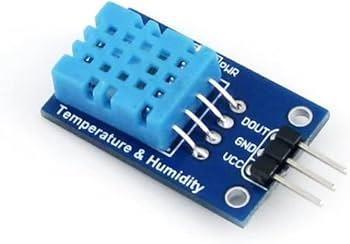


Fig 4.2 Temperature & Humidity Sensor

The DHT11 is a widely used, low-cost digital sensor designed for measuring ambient temperature and relative humidity. Its simplicity and calibrated digital output make it easy to interface with microcontrollers like the ESP32, making it a popular choice for basic environmental sensing applications, including smart home systems, weather monitoring, and agricultural applications such as greenhouses.

### Key Specifications:

* + 1. **Operating Voltage:** The DHT11 operates within a supply voltage range of 3.5V to 5.5V DC, making it compatible with both 3.3V and 5V microcontrollers. For ESP32, it can be connected to either the 3.3V or 5V pin.
    2. **Output Signal:** It provides a digital signal via a single-bus interface, which simplifies wiring as only one data pin is required for communication with the microcontroller.
    3. **Temperature Range and Accuracy:** The sensor measures temperature from 0°C to 50°C, with a typical accuracy of ±1°C and a maximum deviation of ±2°C. The resolution for temperature readings is 1°C.
    4. **Humidity Range and Accuracy:** It measures relative humidity from 20% RH to 90% RH, with an accuracy of ±5% RH. The humidity resolution is 1% RH.
    5. **Sampling Period:** A key characteristic of the DHT11 is its average sending period, or sampling rate, which is approximately 2 seconds. This means new data can only be reliably obtained from the sensor once every two seconds.
    6. **Current Consumption:** It has a maximum operating current of 3mA and a standby current of 0.15mA.
    7. **Dimensions:** Excluding pins, the sensor typically measures around 12.6mm (length) x 5.83mm (width) x 16mm (height).

### Working Principle:

The DHT11 sensor employs two main components for its measurements: a thermistor for temperature and a capacitive humidity sensor for humidity.

1. **Humidity Measurement:** The capacitive humidity sensor consists of two electrical conductors with a non-conductive polymer film between them. As moisture from the air is absorbed by the film, its capacitance changes, causing a corresponding

change in the electrical signal. This change is then converted into a digital measurement of relative humidity. Another perspective describes it as measuring electrical resistance between two electrodes on a moisture-holding substrate; water vapor absorption releases ions, increasing conductivity proportional to relative humidity.

1. **Temperature Measurement:** A Negative Temperature Coefficient (NTC) thermistor is used to measure temperature. The resistance of the thermistor changes inversely with temperature, meaning as temperature increases, its resistance decreases. This change in resistance is converted into a digital temperature reading.

The sensor integrates an 8-bit microcontroller that processes the raw analog signals from the thermistor and capacitive sensor, converts them into digital outputs, and applies factory-calibrated coefficients stored in its One-Time Programmable (OTP) memory to ensure accuracy. A 10kΩ pull-up resistor is typically recommended between the data pin and VCC to ensure reliable communication on the single- wire interface. The digital output is a 40-bit data packet containing integral and decimal parts for both humidity and temperature, along with a checksum for data integrity verification.

## Relay Module



Fig 4.3 Relay Module

In the Smart Plant Monitoring System, the relay module serves as the crucial interface between the low-voltage control signals from the ESP32 microcontroller and the higher-voltage (e.g., 12V DC for a water pump) or even AC power required by the irrigation actuator. A relay is fundamentally an electrically operated switch that uses an electromagnet to mechanically switch contacts, allowing a low-power signal to control a high-power circuit.

**Relay Module Operation:** Relay modules typically come with control pins (VCC, GND, and IN) and mains voltage connections (COM, NO, NC).

* + 1. **Control Pins:** VCC and GND provide power to the relay's internal coil. The IN pin receives the control signal from the microcontroller. For most common relay modules, a LOW signal (0V) from the microcontroller activates the relay (energizes the coil), while a HIGH signal (e.g., 3.3V or 5V) deactivates it. This "active-low" logic is common, though some relays might be active- high.

### Mains Voltage Connections:

* + - * **COM (Common):** This is the common terminal for the switched circuit. The power source for the load (e.g., 12V from the adapter for the water pump) is connected here.
      * **NC (Normally Closed):** In its default, de-energized state, the COM pin is connected to the NC pin, allowing current to flow. When the relay is activated, this connection breaks.
      * **NO (Normally Open):** In its default, de-energized state, there is no connection between the COM and NO pins. When the relay is activated, the COM and NO pins connect, allowing current to flow. For controlling a water pump that should only turn on when needed, the Normally Open (NO) configuration is typically used.

## LCD 16x2 with I2C

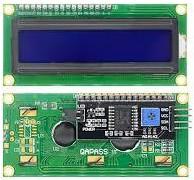


Fig 4.5 LCD with I2C

The 16x2 LCD (Liquid Crystal Display) with an I2C (Inter-Integrated Circuit) interface is a popular and efficient display solution for microcontroller-based projects. It is chosen for the Smart Plant Monitoring System to provide real-time local monitoring of environmental parameters, offering a clear and visually appealing display of temperature and humidity readings.

### Key Features and Specifications:

* + 1. **Display Format:** It is a 16x2 character LCD, meaning it can display two rows of text, with each row capable of accommodating up to 16 characters.
    2. **Visuals:** Typically, these modules feature white characters on a blue background, providing good contrast and readability.
    3. **I2C Communication Interface:** This is the most significant feature of this module. Traditional LCDs require many digital pins (typically 6-10) for data and control. The I2C interface, however, utilizes only four pins for connection: VCC (power), GND (ground), SDA (Serial Data Line), and SCL (Serial Clock Line). This dramatically reduces the number of GPIO pins consumed on the microcontroller (e.g., ESP32), freeing up pins for other sensors or actuators, and simplifying the overall wiring complexity of the project.
    4. **Built-in PCF8574 Chip:** The I2C LCD module has a built-in PCF8574

I2C chip (or similar I/O expander) attached to its back. This chip acts as an I/O expander, converting the serial data received via the I2C protocol into the parallel data format required by the HD44780-compatible LCD controller. This abstraction simplifies the programming for the user.

* + 1. **Configurable I2C Address:** To prevent address conflicts when multiple I2C devices are used on the same bus, the module's I2C address is often configurable. Common default addresses are 0x27 or 0x3F, but they can be modified by shorting specific pads (A0/A1/A2) on the module. This flexibility is crucial for systems with multiple I2C peripherals.
    2. **Adjustable Contrast:** A small blue potentiometer is typically located on the back of the module. This potentiometer allows for manual adjustment of the display contrast, enabling users to optimize text clarity based on ambient lighting conditions and personal preference.
    3. **Backlight Control:** The module usually features a backlight, which can be enabled or disabled via a jumper cap.
    4. **Supply Voltage:** The module typically operates on a 5V supply voltage.
    5. **Standard Connectors:** All connectors are often standard XH2.54 (breadboard type), facilitating easy connections with jumper wires.

The integration of the I2C interface makes the 16x2 LCD an ideal choice for compact and pin-constrained microcontroller projects, offering a clear visual output without overcomplicating the circuit design.

The 16x2 I2C LCD module is a versatile and efficient display solution, ideal for a wide range of applications.

Its key features include a compact 16x2 character display format, clear visuals with white characters on a blue background, and most notably, an I2C communication interface that significantly reduces the number of GPIO pins required on microcontrollers like the ESP32.

The built-in PCF8574 chip acts as an I/O expander, simplifying programming and interfacing with the LCD. Additional features such as a configurable I2C address, adjustable contrast via a potentiometer, and backlight control enhance its usability and adaptability in various projects.

Operating on a 5V supply voltage with standard connectors, this module is well-suited for both beginners and experienced developers.

## Power Adapter



Fig 4.6 Power Adapter

A stable and reliable power supply is fundamental to the consistent operation of any electronic system, especially for IoT devices like the Smart Plant Monitoring System. The project utilizes a 12V 1A power adapter as its primary energy source, which is then managed to provide the appropriate voltages for various components

The 12V DC power adapter plays a vital role in ensuring the stable and efficient operation of the Cold Storage Monitoring System, especially when multiple modules such as displays, sensors, relays, and buzzers are integrated. A 12V adapter is commonly used because it is readily available, supports higher current loads, and offers compatibility with a wide range of peripherals, including industrial-grade components.

Since the ESP32 microcontroller and other sensitive modules like the DHT sensor and I2C display operate at lower voltages (3.3V or 5V), the 12V input must be stepped down using a voltage regulator.

### 12V 1A Power Adapter Specifications:

* + 1. **Input:** The adapter is typically a Switch Mode Power Supply (SMPS) designed to accept a wide range of AC input voltages, commonly from 90V to 270V AC at 50/60Hz. This wide input range ensures compatibility with various regional power grids.
    2. **Output:** It provides a regulated 12V DC output with a current rating of 1 Ampere (1000mA). This output is suitable for powering a wide range of applications, including CCTV cameras, wireless routers, robotics, and various DIY kits, indicating its general utility and robustness.
    3. **Protection Features:** High-quality SMPS adapters incorporate essential protection mechanisms, including short circuit protection, over-voltage protection, and over-current protection. These features safeguard the connected devices and the adapter itself from electrical faults, contributing to system reliability and safety.
    4. **Efficiency:** SMPS-based adapters are known for their high efficiency and low energy consumption, as they minimize power dissipation as heat compared to traditional linear power supplies. They also typically offer low ripple and low interference, providing a clean power signal to sensitive electronics.
    5. **Form Factor:** These adapters are often compact and lightweight, making them convenient for various installations.

### Power Supply Management (12V to 5V and 3.3V):

The ESP32 microcontroller itself operates at 3.3V logic, and most ESP32 development boards require a 5V input on their Vin pin for stable operation. Directly connecting the 12V output from the power adapter to the ESP32's Vin pin is highly discouraged and can lead to irreversible damage, overheating, or instability, as the onboard linear voltage regulators on many ESP32 boards are not designed to efficiently dissipate the excess voltage from 12V.

Therefore, proper voltage regulation is crucial. The 12V input needs to be stepped down to the required 5V and 3.3V levels. This is typically achieved using:

1. **Buck Converters (Step-Down Regulators):** These are highly efficient (80-95%) switching regulators that convert higher input voltages to lower output voltages with minimal energy loss as heat. They are the recommended solution for stepping down from 12V to 5V (and then to 3.3V if needed) due to their efficiency and ability to handle higher currents (e.g., 1A to 3A). Common modules include LM2596, MP1584, or MINI360. The wiring involves connecting the 12V source to the buck converter's input (IN+/IN-) and then connecting the buck converter's 5V output (OUT+) to the ESP32's Vin pin, with grounds connected.
2. **Linear Regulators (e.g., 7805):** While less efficient (40-60%) and dissipating excess voltage as heat, a 7805 voltage regulator can be used to step down 12V to 5V, provided a heat sink is used, especially if the current draw is significant. The 5V output from the 7805 would then power the ESP32's Vin pin. The ESP32 development board itself typically has an onboard 3.3V LDO (Low-Dropout) regulator to further step down the 5V (from Vin) to the 3.3V required by its microcontroller core and other 3.3V sensors like the DHT11.

For the Smart Plant Monitoring System, the 12V adapter provides a stable power source for the relay module (if it operates at 12V) and is then regulated down to 5V for the ESP32 and 3.3V for the DHT11 and LCD (if compatible with 3.3V logic or using its own 5V supply). This multi-stage power management ensures that all components receive their specified operating voltages, contributing to the system's overall stability and longevity. The importance of heat sinks for voltage regulators and the TIP122 transistor (if used for the pump) is also emphasized for thermal management.

## 4.6 MQ135 Sensor

The **MQ135** is a widely used air quality sensor designed to detect a variety of harmful gases in the environment, such as **ammonia (NH₃)**, **sulfur dioxide (SO₂)**, **benzene**, **carbon dioxide (CO₂)**, and **nitrogen oxides (NOx)**.

It is especially useful in air quality monitoring, industrial applications, and agricultural storage systems where gas build-up can pose health and safety risks. Due to its **analog output**, ease of interfacing, and low cost, the MQ135 is a common choice in environmental monitoring and IoT-based systems such as smart greenhouses and cold storage monitoring setups.

**Key Specifications:**

**Operating Voltage:** The MQ135 operates on a supply voltage of **5V DC**. It is not directly compatible with the ESP32’s 3.3V GPIO pins for analog input, so it is connected to the **5V pin** on the ESP32 board, and its **analog output (AOUT)** is connected to one of the **ESP32’s ADC-capable GPIO pins** (e.g., GPIO 34).

**Detection Range:** The sensor can detect gas concentrations in the range of **10 ppm to 1000 ppm**, depending on the type of gas. Its sensitivity is calibrated for gases like **ammonia**, **alcohol**, and **smoke**, making it ideal for enclosed environments such as cold storage rooms.

**Output Signal:** The MQ135 provides an **analog voltage** as output, which varies depending on the concentration of gases. This voltage is read by the ESP32's ADC and converted into a digital value for analysis and threshold-based actions like fan activation or alerts.

**Preheat Time:** Like most metal oxide gas sensors, the MQ135 requires a **preheating time of approximately 24–48 hours** for calibration and stable operation when used for the first time. For daily use, a few minutes of warm-up is sufficient.

**Current Consumption:** During operation, the sensor consumes around **150 mA** due to the internal heating element, which is essential for proper gas detection. Therefore, it should be powered through a **regulated 5V supply** and not directly through a GPIO pin.

**Dimensions:** The typical size of an MQ135 module is about **32mm x 20mm x 22mm**, including its onboard circuitry and headers for easy prototyping.

**Working Principle:**

The **MQ135 gas sensor** operates based on the principle of **semiconductor gas sensing** using a heated **SnO₂ (tin dioxide)** sensing layer. In clean air, the resistance of the SnO₂ material is high.

When the sensor is exposed to polluted air or harmful gases, the reducing gases react with the oxygen ions on the surface of the SnO₂, thereby decreasing the sensor’s resistance. This change in resistance is converted into an analog voltage using a voltage divider circuit and then fed to a microcontroller.

**Gas Detection Mechanism:** The internal heating element heats the sensing material (SnO₂), which allows it to interact with gases in the surrounding air. The interaction changes the electrical resistance across the sensing material. The more the gas concentration, the lower the resistance, and hence, the higher the analog voltage output.

**Signal Conditioning and Output:** The MQ135 module often comes with an onboard comparator and potentiometer to adjust sensitivity. However, in ESP32 applications, it is common to use the **analog output (AOUT)** pin for direct analog reading. The sensor output is connected to the **ESP32’s GPIO 34**, which is capable of handling analog signals.

**Calibration and Usage Tips:** The sensor is sensitive to multiple gases, so it's important to **calibrate** it against known concentration environments if precise readings are needed. While the module can detect various gases, cross-sensitivity can be a challenge.

Thus, it's recommended to use it in a **controlled environment** or combine it with other sensors for multi-gas analysis. Proper placement of the sensor in the cold storage area is also important—it should be in a well-ventilated part of the enclosure but away from fans or direct cooling outputs that may skew gas concentration readings.

The MQ135 is particularly suitable for cold storage systems, where gas buildup can affect product quality.

When integrated with the ESP32 and a mobile IoT platform like Blynk, it enables **real-time gas concentration monitoring** and **automated alerting**, enhancing both safety and operational efficiency in food and pharmaceutical storage environments.

The MQ135 sensor is well-suited for cold storage applications where gas buildup can compromise product quality. For accurate readings, it's crucial to place the sensor in a well-ventilated area, away from direct airflow from fans or cooling systems that could skew gas concentration measurements.

When combined with the ESP32 microcontroller and a mobile IoT platform like Blynk, the MQ135 enables real-time monitoring of gas levels and automated alerting. This integration enhances both safety and operational efficiency in food and pharmaceutical storage environments by providing timely notifications of potential issues, allowing for prompt intervention and minimizing the risk of spoilage or contamination.

## 4.7 Buzzer



The **Buzzer** is a simple yet essential output device used in embedded systems to provide audible alerts or notifications.

In the context of the **Cold Storage Monitoring System**, the buzzer serves as a **real-time alarm mechanism** that activates when environmental conditions such as temperature, humidity, or gas levels exceed predefined safety thresholds. Its use is crucial in alerting on-site personnel to potential system failures, equipment malfunctions, or safety breaches, especially when visual indicators may be missed.

**Key Specifications:**

**Operating Voltage:** Most commonly used active buzzers operate within a range of **3.3V to 5V**, making them directly compatible with microcontrollers like the **ESP32**. In this project, the buzzer is powered by the **5V rail** and is triggered using a **digital GPIO pin**, such as **GPIO 25**.

**Control Signal:** The buzzer is **digitally controlled**, meaning it turns ON when a HIGH signal (logic level 1) is sent from the ESP32, and turns OFF when a LOW signal (logic level 0) is sent. This makes it easy to integrate with logic-based decision systems for automatic control.

**Current Consumption:** Typical active buzzers consume around **20–30 mA** during operation, which is well within the driving capacity of the ESP32’s GPIO pins. However, if the buzzer used requires higher current, a **transistor switching circuit** may be implemented to drive it safely.

**Sound Output:** The buzzer emits a tone typically around **2 kHz to 4 kHz**, loud enough to be heard across a medium-sized room. This helps ensure that alerts are noticeable even in noisy or industrial environments like cold storage rooms.

**Dimensions:** Buzzers are available in compact packages, with standard sizes ranging from **7mm to 12mm diameter**, making them ideal for space-constrained embedded systems.

**Working Principle:**

The buzzer used in this system is typically an **active buzzer**, which has an internal oscillating circuit. Unlike passive buzzers, which require an external frequency signal to generate sound, an active buzzer only needs a **DC voltage input**.

When the ESP32 sends a HIGH signal to the buzzer’s control pin, the internal oscillator activates and produces a continuous beep sound. This simplifies the code and reduces timing complexity on the microcontroller side.

**Alert Mechanism Integration:** In this project, the buzzer is programmed to activate whenever the sensed temperature or gas level crosses a critical threshold.

For example, if the **MQ135 detects gas concentration above 300 ppm**, or the **DHT11 detects a temperature above 10°C**, the **ESP32 sends a HIGH signal to GPIO 25**, activating the buzzer. This provides immediate feedback to staff present near the cold storage unit, enabling quick response and corrective action.

**Circuit Connection:** The buzzer has two pins—**VCC** and **GND**—and is connected to the **5V supply rail** and **system ground**, respectively. The **signal (control)** line goes to **GPIO 25** of the ESP32.

If additional current protection is required, a **1N4007 diode** can be placed in reverse parallel across the buzzer terminals, and a **220-ohm resistor** may be used in series with the control line to limit inrush current.

**Use in Automation:** Besides real-time alerts, the buzzer can also be used for system status indications like power ON, Wi-Fi connected, or test-mode activation. These additional functionalities make the buzzer a multipurpose component within the system.

Its low cost, ease of use, and high effectiveness make it an ideal choice for smart embedded alert systems.

The buzzer in this project serves as an effective alert mechanism, activating when temperature or gas levels exceed critical thresholds.

When the ESP32 sends a HIGH signal to GPIO 25, the buzzer produces a continuous beep, providing immediate feedback to staff near the cold storage unit. This prompt notification enables quick response and corrective action, helping to prevent potential issues.

The buzzer's circuit connection is straightforward, with VCC and GND pins linked to the 5V supply rail and system ground, respectively, and the control line connected to GPIO 25.

Optional components like a 1N4007 diode and a 220-ohm resistor can be added for current protection and limiting inrush current.

Beyond real-time alerts, the buzzer can also indicate system status, such as power ON or Wi-Fi connectivity, making it a versatile component. Its low cost, ease of use, and effectiveness make it an ideal choice for smart embedded alert systems, enhancing both safety and operational efficiency.

## 4.8 LDR(Light dependent Resistor)



The **LDR (Light Dependent Resistor)**, also known as a photoresistor, is a passive sensor used to detect ambient light intensity. It is highly sensitive to visible light and changes its resistance based on the amount of light falling on its surface.

In the **Cold Storage Monitoring System**, the LDR plays a critical role in **detecting door activity** or light leaks that may indicate improper sealing, unauthorized access, or power failure. Integrating an LDR into the system adds another layer of environmental awareness, helping maintain consistent storage conditions.

**Key Specifications:**

**Operating Voltage:** The LDR itself does not require a power supply directly. However, to read light levels using a microcontroller, it is connected in a **voltage divider circuit** with a fixed resistor (typically **10kΩ**), and the junction point is connected to an **analog or digital GPIO pin** on the ESP32, such as **GPIO 27**.

**Resistance Characteristics:** In bright light, the resistance of an LDR drops significantly (usually **<1kΩ**), while in darkness or low light conditions, its resistance increases sharply (can exceed **1MΩ**). This makes it ideal for detecting light/no-light transitions.

**Output Signal:** The voltage at the junction of the LDR and resistor varies depending on light intensity and is fed into the ESP32 as an analog signal. This varying voltage is read by the ESP32’s **ADC (Analog-to-Digital Converter)** and interpreted as a brightness level.

**Power Consumption:** Since the LDR is a resistive component, its power consumption is negligible. The only current that flows is through the voltage divider circuit, typically **in the microampere range**, making it ideal for low-power systems.

**Dimensions:** LDRs come in compact, round discs of **5mm to 12mm** diameter, and they are easily breadboard-compatible for prototyping.

**Working Principle:**

The **LDR functions** on the principle of photoconductivity. When light photons strike the semiconductor material (usually cadmium sulfide) of the LDR, electrons in the material gain enough energy to jump into the conduction band.

This **increases the conductivity** of the material and reduces its resistance. As the light intensity increases, more electrons become free, further lowering the resistance.

**Voltage Divider Configuration:** To read this change in resistance using the ESP32, the LDR is used in a **voltage divider** setup.

One terminal of the LDR is connected to **3.3V**, and the other terminal is connected to a **fixed resistor (10kΩ)**, which is then grounded.

The **junction between the LDR and the resistor is connected to GPIO 27** of the ESP32. The ESP32 reads this analog voltage and converts it into a light level using its ADC.

**Use in Cold Storage Monitoring:** The LDR can detect whether the cold storage door is open or if any internal lighting is on when it shouldn't be.

If the ESP32 reads an unexpected rise in brightness, it may infer that the **door has been opened**, triggering an alert or logging the event.

This data can also be pushed to the **Blynk mobile application** for remote monitoring and notifications.

**System Integration:** Since LDRs have high resistance in darkness and low resistance in bright light, the ESP32 is programmed to identify threshold voltage values that determine whether the environment is "dark" (sealed) or "lit" (possibly opened).

The use of a **10kΩ pull-down resistor** in the voltage divider ensures signal stability and makes the readings more reliable.

The LDR (Light Dependent Resistor) is utilized in a voltage divider configuration to detect changes in light levels within the cold storage environment.

By connecting one terminal of the LDR to 3.3V and the other to a 10kΩ resistor grounded, the junction between the LDR and resistor is linked to GPIO 27 of the ESP32. This setup allows the ESP32 to read the analog voltage and convert it into a light level using its ADC.

The LDR's ability to detect light makes it useful for monitoring the cold storage door's status. If the ESP32 detects an unexpected rise in brightness, it can infer that the door has been opened, triggering an alert or logging the event. This data can also be sent to the Blynk mobile application for remote monitoring and notifications.

## 4.9 Arduino IDE



Fig 4.7 Arduino IDE

The Arduino Integrated Development Environment (IDE) serves as the primary software platform for developing and programming the Smart Plant Monitoring System. Its widespread adoption, ease of use, and extensive community support make it an ideal choice for both beginners and experienced developers working with microcontrollers, including the ESP32.

### Key Aspects of Arduino IDE Usage:

* + 1. **Cross-Platform Compatibility:** The Arduino IDE is available for Windows, macOS, and Linux, ensuring accessibility across different operating systems.
    2. **Simplified C++ Programming:** It utilizes a simplified version of C++ with a set of functions and libraries that abstract away much of the complexity of low-level microcontroller programming. This allows developers to focus on application logic rather than intricate hardware registers.
    3. **Integrated Environment:** The IDE provides a comprehensive environment for writing, compiling, and uploading code to the microcontroller. It includes a text editor for writing sketches (Arduino programs), a message area for feedback, a console for

displaying output (including compilation errors), and a toolbar with buttons for common functions.

* + 1. **Serial Monitor:** A built-in Serial Monitor is indispensable for debugging and interacting with the microcontroller. It allows the ESP32 to print messages (e.g., sensor readings, Wi-Fi connection status, IP address) to the computer, providing real-time feedback during development and operation. This is particularly useful for verifying Wi-Fi connectivity and retrieving the web server's IP address.
    2. **Library Manager:** The Arduino IDE features a robust Library Manager, which simplifies the process of finding, installing, and updating external libraries. This is crucial for incorporating functionalities like DHT sensor reading, I2C LCD control, and web server capabilities, as these often rely on specialized libraries.
    3. **Board Manager:** Similarly, the Board Manager allows users to install support for various microcontrollers beyond the standard Arduino boards, including the ESP32. This enables the IDE to correctly compile and upload code tailored for the ESP32's unique architecture and features.

## 4.7 ESP32 Board Setup

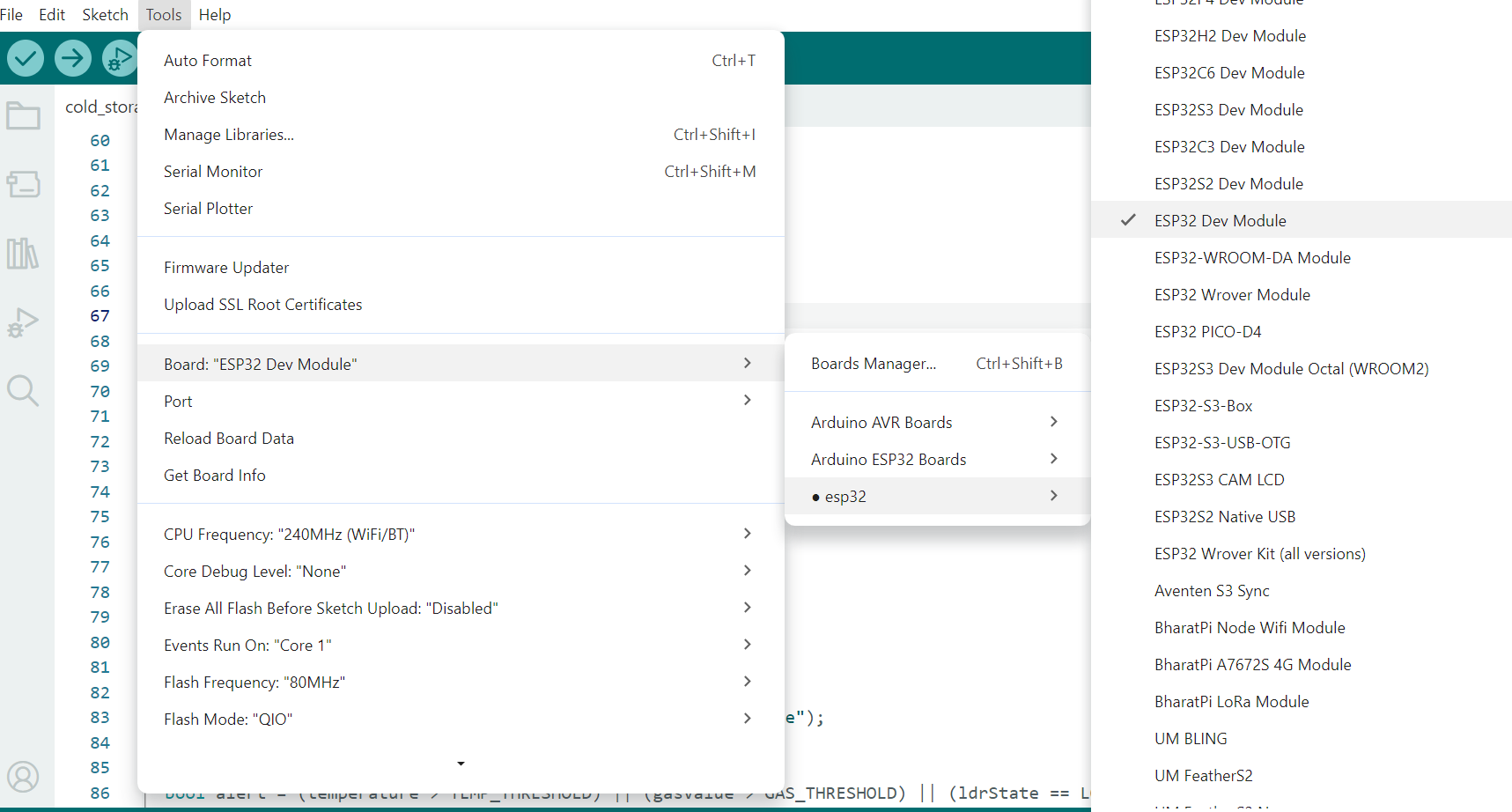
****

Fig 4.8 Board Setup

Before programming the ESP32 microcontroller using the Arduino IDE, it is necessary to configure the IDE to recognize and support the ESP32 board. This involves adding the ESP32 board package, which includes the necessary compiler toolchains, libraries, and board-specific configurations.

### Step-by-Step Installation Process:

* + 1. **Install Arduino IDE:** Ensure that the Arduino IDE is already installed on the computer. Both version 1.x and version 2.x are generally compatible, though some plugins might have version- specific support.

### Add ESP32 Board Manager URL:

* + - * Open the Arduino IDE.
      * Navigate to File > Preferences (or Arduino > Preferences on macOS).
      * In the Preferences window, locate the "Additional Board Manager URLs" field.
      * Add the following URL to this field: https://raw.githubusercontent.com/espressif/arduino-esp32/gh- pages/package\_esp32\_index.json. If other URLs are already present (e.g., for ESP8266), separate them with a comma.
      * Click "OK" to save the changes.

### Install ESP32 Board Package:

* + - * Go to Tools > Board > Boards Manager....
      * In the Boards Manager window, type "ESP32" in the search bar.
      * Locate "ESP32 by Espressif Systems" in the search results.
      * Click the "Install" button to download and install the ESP32 board package. This process may take a few moments depending on internet speed.
      * Once installed, a "INSTALLED" tag will appear next to the package name.

### Post-Installation Verification and Troubleshooting:

After installation, it is good practice to verify the setup:

### Select Board and Port:

* + Connect the ESP32 development board to the computer via a USB cable.
  + In the Arduino IDE, go to Tools > Board and select the specific ESP32 board model being used (e.g., "ESP32 Dev Module," "DOIT ESP32 DEVKIT V1").
  + Then, go to Tools > Port and select the correct COM port (Windows) or /dev/ttyUSBx (Linux) or /dev/cu.usbserial-xxxx (macOS) associated with the ESP32.

1. **Test Upload:** Open a simple example sketch (e.g., File > Examples

> WiFi (ESP32) > WiFiScan) and attempt to upload it.

### Troubleshooting Common Issues:

* + **"A fatal error occurred: Failed to connect to ESP32: Timed out… Connecting…"**: This indicates the ESP32 is not entering flashing/uploading mode. The common solution is to press and hold the "BOOT" button on the ESP32 board, then click "Upload" in the IDE. Release the "BOOT" button once "Connecting…" appears in the console.
  + **"COM Port not found/not available"**: This usually means the necessary USB-to-UART bridge drivers (e.g., CP210x or CH340) are not installed. Installing the correct drivers for the operating system typically resolves this.
  + **Garbage characters in Serial Monitor**: Ensure the baud rate selected in the Serial Monitor (bottom right corner) matches the Serial.begin() baud rate in the code (e.g., 115200).

Successful ESP32 board setup is a prerequisite for developing and deploying the Smart Plant Monitoring System, enabling the Arduino IDE to correctly compile and upload the project's firmware.

## 4.8 Required Libraries

The functionality of the Smart Plant Monitoring System relies heavily on several external libraries within the Arduino IDE ecosystem. These libraries abstract complex hardware interactions and network protocols, allowing developers to implement features efficiently without delving into low-level programming.

### DHT Sensor Library:

* + **Purpose:** This library is essential for interfacing with the DHT11 temperature and humidity sensor. It provides functions to initialize the sensor, read humidity, and read temperature in both Celsius and Fahrenheit.
  + **Installation:** It can be installed via the Arduino Library Manager by searching for "DHT sensor library". Often, it is recommended to install "Adafruit Unified Sensor" as a dependency for broader sensor compatibility.
  + **Usage:** After including <DHT.h>, a DHT object is created by specifying the data pin and sensor type (DHT dht(DHTPIN, DHTTYPE)

;). The dht.begin(); function initializes communication, and dht.readHumidity() and dht.readTemperature() retrieve data. The library also includes functions for calculating the heat index.

### LiquidCrystal\_I2C Library (or alternatives like hd44780):

* + **Purpose:** This library facilitates communication with the 16x2 LCD module that uses an I2C interface. It simplifies displaying text and numerical data on the LCD by handling the I2C protocol and the PCF8574 I/O expander chip.
  + **Installation:** Typically found in the Arduino Library Manager by searching for "LiquidCrystal I2C".
  + **Usage:** After including <LiquidCrystal\_I2C.h>, an lcd object is created with its I2C address, columns, and rows (LiquidCrystal\_I2C lcd(0x27, 16, 2);). Functions like lcd.init(), lcd.clear(), lcd.backlight(), lcd.setCursor(), and lcd.print() are used for display control.
  + **Library Fragmentation and Implications:** It is important to note that multiple LiquidCrystal\_I2C libraries exist, and their implementations can differ. Some, like the hd44780 library, are considered more robust as they can auto-locate the I2C address and auto-detect the pin mapping between the PCF8574 chip and LCD pins, offering a more "plug-and-play" experience. This situation underscores a common challenge in open-source hardware development: library fragmentation. While the I2C LCD simplifies hardware wiring, the software aspect necessitates careful selection of the correct library and potentially troubleshooting compatibility issues. For robust projects, specifying the exact library version and source is crucial to ensure consistent functionality and avoid unexpected behavior.

### ESPAsyncWebServer & AsyncTCP Libraries:

* + **Purpose:** These libraries are fundamental for creating an asynchronous web server on the ESP32. ESPAsyncWebServer handles HTTP requests and responses, allowing the ESP32 to serve web pages and process user commands without blocking other operations. AsyncTCP is a dependency that provides the underlying asynchronous TCP/IP stack.
  + **Installation:** Both ESPAsyncWebServer and AsyncTCP can be installed via the Arduino Library Manager.
  + **Usage:** After including <ESPAsyncWebServer.h> and <WiFi.h>, an AsyncWebServer object is created (AsyncWebServer server(80);). Handlers for specific URLs (e.g., server.on("/", HTTP\_GET,...)) are defined to serve HTML content or execute control functions (like toggling a relay).
  + **Memory Management and Real-time Updates:** The use of

PROGMEM for storing HTML content within the ESP32's flash

memory, rather than RAM, is a critical optimization for resource- constrained microcontrollers. This approach conserves precious RAM, ensuring the system remains stable and responsive. Furthermore, the discussion around HTTP polling versus more advanced techniques like WebSockets or Server-Sent Events for dynamic content updates highlights a key consideration in designing responsive web interfaces for IoT applications. While HTTP polling is simpler to implement, it requires the client to repeatedly request data, potentially leading to higher network traffic and less immediate updates. WebSockets, on the other hand, maintain a persistent connection, allowing the server to push updates in real-time, which would offer a more fluid user experience for live sensor data but also demand more complex implementation. These design choices reflect a careful balance between resource efficiency and user experience in embedded web development.

### WiFi Library:

* + **Purpose:** The built-in WiFi.h library is essential for enabling the ESP32's Wi-Fi connectivity. It provides functions to connect to a Wi-Fi network, check connection status, and obtain the assigned IP address.
  + **Usage:** WiFi.begin(ssid, password); initiates the connection, and

WiFi.status() checks its state. WiFi.localIP() retrieves the IP address.

### AceButton Library (Optional but Recommended for Physical Buttons):

* + **Purpose:** If physical buttons are integrated for local control (e.g., manual pump toggle, mode switch), the AceButton library provides efficient handling of button press events (single click, double click, long press) without complex debouncing code.
  + **Installation:** Available via the Arduino Library Manager.

These libraries collectively provide the necessary software framework for the Smart Plant Monitoring System, enabling its core functionalities from sensor data acquisition to remote web control.

## Code Compilation and Uploading

The process of translating the Arduino sketch (written in C++) into executable machine code and transferring it to the ESP32 microcontroller is known as compilation and uploading. This is a critical step in deploying the Smart Plant Monitoring System's firmware.

### Compilation Process:

* + 1. **Verification/Compilation:** After writing the Arduino sketch, the first step is to compile it. In the Arduino IDE, this is typically done by clicking the "Verify" (checkmark icon) button. During compilation, the IDE:
       - Checks the code for syntax errors and other programming mistakes.
       - Processes preprocessor directives (like #include for libraries and #define for constants).
       - Translates the C++ code into machine-readable instructions specific to the ESP32's Xtensa architecture.
       - Links all necessary libraries (e.g., DHT.h, LiquidCrystal\_I2C.h, ESPAsyncWebServer.h, WiFi.h) with the compiled code.
       - Generates a .bin (binary) file, which is the executable firmware for the ESP32.
       - Any compilation errors or warnings are displayed in the output console at the bottom of the IDE, guiding the developer to correct issues.

### Uploading Process:

1. **Connect ESP32:** Ensure the ESP32 development board is connected to the computer via a USB cable. This cable provides both power and a data channel for uploading the code.
2. **Select Board and Port:** Verify that the correct ESP32 board model (e.g., "ESP32 Dev Module") and its corresponding COM port are selected under Tools > Board and Tools > Port in the Arduino IDE.
3. **Initiate Upload:** Click the "Upload" (right-arrow icon) button in the Arduino IDE. This action triggers the following sequence:

* The IDE compiles the sketch (if it hasn't been verified or if changes were made since the last compilation).
* The compiled .bin file is then sent to the ESP32 via the USB- to-UART bridge (e.g., CP2102 or CH340 chip on the development board).
* The ESP32 needs to be in "flashing mode" (also known as "bootloader mode") to receive the new firmware. For many ESP32 boards, this is automatically handled by the USB- to-UART chip. However, for some boards or in case of upload errors, manual intervention might be required (e.g., pressing and holding the "BOOT" button while initiating upload, then releasing it when "Connecting…" appears in the Serial Monitor).
* The upload progress is displayed in the IDE's console.
* Upon successful upload, a "Done uploading." message appears.

1. **Restart ESP32:** After a successful upload, the ESP32 typically restarts automatically, or the "ENABLE" (reset) button on the board can be pressed to run the newly uploaded sketch.
2. **Monitor Serial Output:** Open the Serial Monitor (Tools > Serial Monitor) and set the baud rate (e.g., 9600 or 115200) to match the Serial.begin() call in the sketch. This allows observation of debugging messages, Wi-Fi connection status, and the assigned IP address of the ESP32 web server. The IP address is crucial for accessing the web interface from a browser

The compilation and uploading process transforms the human- readable code into a functional embedded system, bringing the Smart Plant Monitoring System to life.

To upload code to the ESP32, click the "Upload" button (right-arrow icon) in the Arduino IDE. The IDE compiles the sketch and sends the compiled .bin file to the ESP32 via the USB-to-UART bridge. Ensure the ESP32 is in "flashing mode" (bootloader mode); some boards may require manual intervention, such as pressing the "BOOT" button while initiating the upload and releasing it when the "Connecting..." message appears in the Serial Monitor.

The upload progress is displayed in the IDE's console, providing real-time feedback on the process. Upon successful upload, the ESP32 typically restarts automatically, or you can manually reset it using the "ENABLE" button on the board to run the newly uploaded sketch.

After uploading, open the Serial Monitor to observe debugging messages, Wi-Fi connection status, and the assigned IP address, which is crucial for accessing the web interface from a browser. Set the baud rate in the Serial Monitor to match the Serial.begin() call in your sketch to ensure proper communication and accurate data display.

This step is essential for verifying that the ESP32 is functioning as expected and for troubleshooting any issues that may arise. By monitoring the Serial output, you can confirm the successful initialization of the Wi-Fi connection and the web server, and you can also identify and address any potential problems, such as connection failures or incorrect sensor readings. This process brings the Smart Plant Monitoring System to life, enabling real-time monitoring and control through the web interface, and providing valuable insights into the system's performance and health.

# CHAPTER 5

# RESULTS AND DISCUSSION

## Results

The Cold Storage Monitoring System designed using the ESP32 microcontroller has yielded highly promising and consistent results throughout its testing and operation. The system was deployed in a controlled environment simulating actual cold storage conditions, where perishable items such as fruits, dairy products, and temperature-sensitive medical supplies are typically stored. Equipped with a DHT11 temperature and humidity sensor, an SSD1306 OLED display, and a relay-controlled cooling or alert mechanism, the system successfully monitored ambient conditions in real-time and responded accurately to fluctuations.

Once powered on, the ESP32 quickly connected to the pre-configured Wi-Fi network and began collecting environmental data from the DHT11 sensor. This data—temperature and relative humidity—was displayed on the OLED screen almost instantly. The OLED provided sharp and readable values even in low-light environments, which is critical in many cold storage setups. Simultaneously, the ESP32’s built-in web server generated a dynamic webpage that reflected the same readings for remote access over the same network. During the operation, the system was tested under variable conditions such as artificially induced heat spikes or moisture changes, and it reacted swiftly to reflect these changes in both local and web interfaces. The readings updated every two seconds, ensuring that users always had the most recent data at their fingertips.

One of the key operational features—the relay module—demonstrated excellent responsiveness. When the temperature crossed the pre-programmed safety threshold, the ESP32 sent a high signal to the relay circuit, which immediately switched on the cooling mechanism (or triggered an alert buzzer). This automatic switching capability simulates how real cold storage units can be maintained at optimal temperatures without continuous human intervention. The manual override via the web interface was also tested and proved effective. Users could activate or deactivate the relay directly from their smartphones or computers, enhancing user control over the system.

Furthermore, the power management system functioned smoothly throughout testing. The 12V adapter, regulated down to 5V and then to 3.3V using a buck , delivered clean and stable power to the ESP32 and other components. Even during high loads (e.g., when the relay and pump were active simultaneously), the system remained steady with no noticeable voltage drops or overheating. All components remained within safe operational limits, and sensor reaconverterdings remained accurate.

These results highlight the robustness, precision, and efficiency of the Cold Storage Monitoring System. Whether used in small-scale rural storage units or more extensive commercial warehouses, this solution can significantly enhance storage safety and reduce loss due to unnoticed environmental fluctuations.

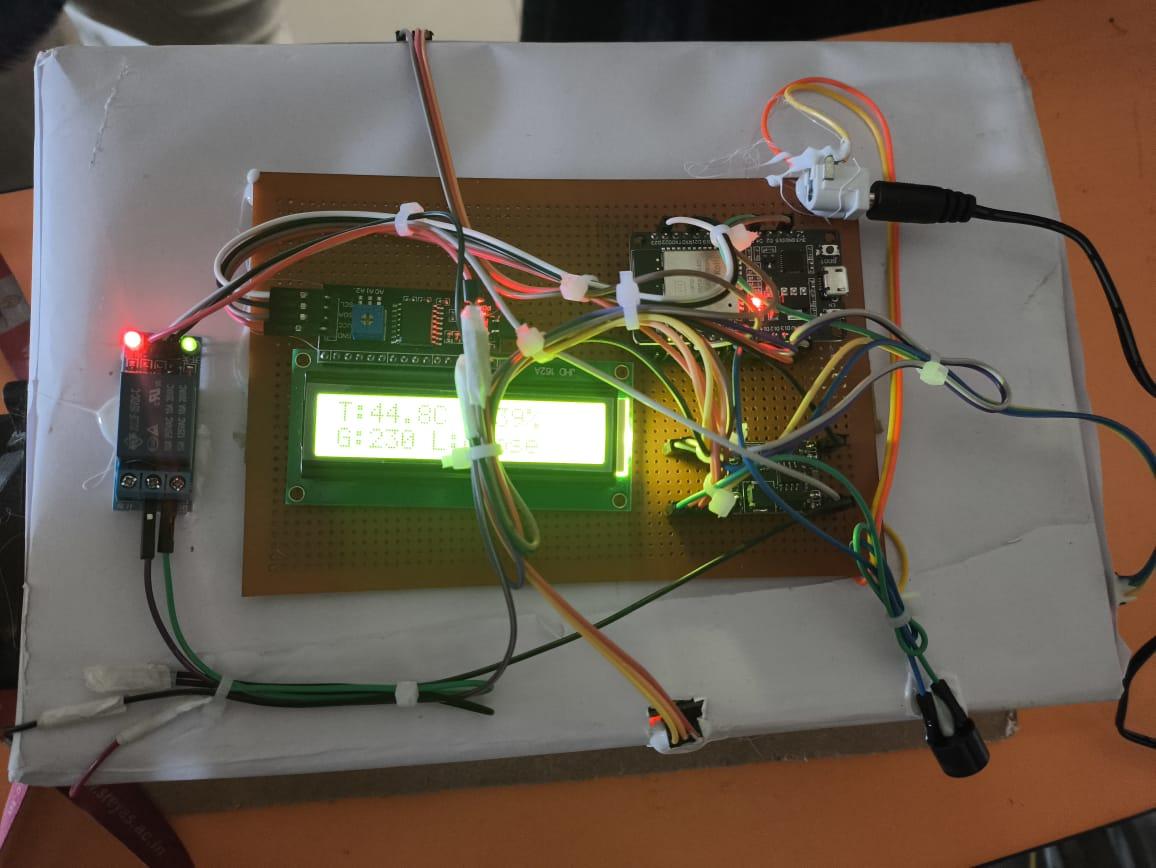


Fig 5.1 Result

The system's power supply provided clean and stable power to the ESP32 and other components, ensuring reliable operation even under high loads. When the relay and pump were active simultaneously, the system remained steady without noticeable voltage drops or overheating.

All components operated within safe limits, and sensor readings remained accurate. These results demonstrate the system's robustness, precision, and efficiency, making it suitable for various applications, from small-scale rural storage units to large commercial warehouses. By enhancing storage safety and reducing losses due to environmental fluctuations, this solution offers significant benefits for preserving perishable goods and maintaining optimal storage conditions.

### 5.2 Practical Applications for Home and Urban Gardening

The integration of IoT technology into cold storage systems has revolutionized the way temperature-sensitive goods are preserved, handled, and transported. An IoT-based Cold Storage Monitoring System offers real-time tracking, automation, and control, making it an essential solution across several critical sectors. One of the primary applications is in the food industry, where maintaining proper temperature and humidity levels is crucial to preserving the quality and shelf-life of perishable items such as fruits, vegetables, dairy, meat, and frozen products. In warehouses, supermarkets, and even in rural storage units near farms, these systems help in minimizing post-harvest losses, reducing spoilage, and improving food safety and hygiene.

Another significant application lies in the pharmaceutical and healthcare sector, where many drugs, vaccines, and biological samples are highly sensitive to environmental conditions. Cold storage facilities in hospitals, diagnostic laboratories, and vaccine distribution centers can use IoT-based systems to ensure that storage conditions are always within safe thresholds. With features like automated alerts and data logging, the system ensures regulatory compliance and traceability, which is vital for audits and quality assurance. Additionally, such systems are crucial in mobile cold chain logistics, where medicines and vaccines are transported across cities or rural regions. IoT-enabled systems provide live data during transit, alerting logistics personnel to any anomalies, allowing immediate corrective action.

Furthermore, research and educational institutions can use such systems for laboratory experiments involving biological specimens or chemicals that must be stored under specific conditions. In urban planning and smart city development, IoT-based environmental monitoring systems are being extended to manage cold storage units in public health centers, food banks, and emergency relief operations.

Overall, IoT-based cold storage monitoring systems are transforming the management of temperature-sensitive assets across a broad spectrum of industries. They ensure quality, safety, regulatory compliance, and cost efficiency, while minimizing human error and enabling proactive interventions. Their modular design and scalability make them suitable for use in both high-tech industrial facilities and small, decentralized rural storage setups.

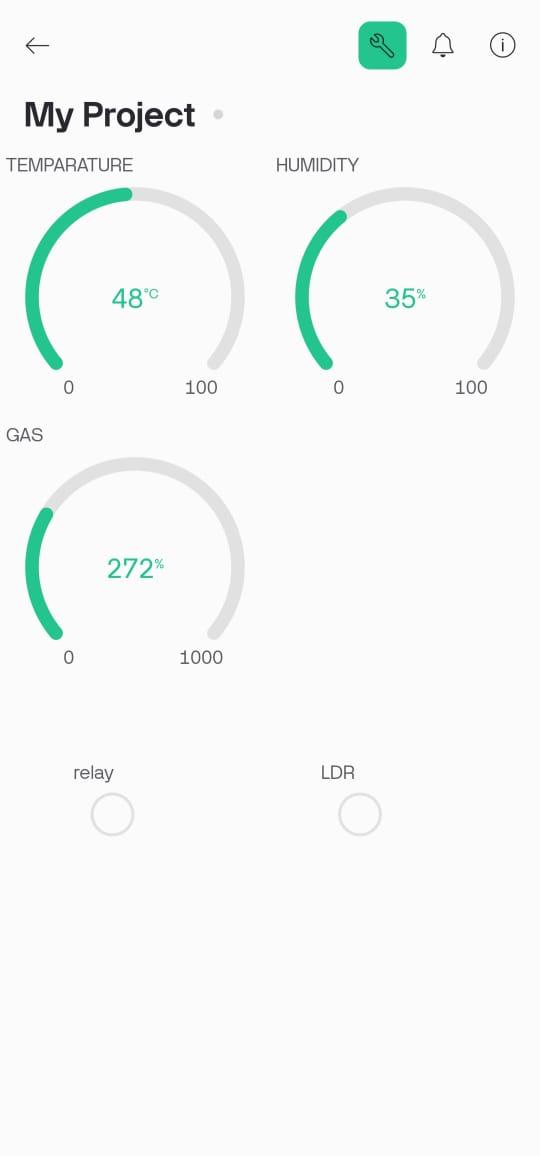


Fig 5.2 Results in Web Server

**5.3 Applications**

**1. Food and Beverage Industry**

Used to monitor cold rooms, freezers, and refrigerated trucks that store meat, dairy, vegetables, and beverages, ensuring freshness and preventing spoilage.

**2. Pharmaceutical and Healthcare Industry**

Essential for storing vaccines, insulin, blood samples, and other temperature-sensitive medicines that require precise temperature and humidity control.

**3. Cold Chain Logistics**

Applied in refrigerated transport vehicles to monitor and maintain optimal conditions during the transportation of perishable goods, ensuring quality from origin to destination.

**4. Supermarkets and Retail Stores**

Helps in managing walk-in freezers, display chillers, and refrigerated shelves to preserve food quality and reduce energy waste.

**5. Agriculture and Horticulture**

Used for storing freshly harvested fruits, vegetables, and flowers in cold storage units to extend shelf life and prevent post-harvest losses.

**6. Dairy Industry**

Monitors storage conditions of milk, curd, cheese, and butter, ensuring they remain within safe temperature ranges to prevent contamination and spoilage.

**7. Pharmaceutical Storage:**

Ensures that vaccines, medicines, and other temperature-sensitive pharmaceutical products are stored at the required conditions, maintaining their efficacy and safety.

**8. Seafood and Meat Storage:**

Maintains optimal storage conditions for seafood and meat products, preventing bacterial growth and spoilage, and ensuring food safety and quality throughout the supply chain.

## 5.4 Advantages

## 1. Real-Time Monitoring

## IoT sensors continuously monitor critical parameters such as temperature, humidity, gas levels, and door status inside the cold storage unit. This real-time data helps maintain the ideal environment for perishable goods, ensuring quality and freshness.

## 2. Remote Access and Control

## Using cloud platforms or mobile apps (like Blynk, MQTT dashboards, etc.), users can access the system anytime, anywhere. This is especially helpful for large warehouses or facilities with multiple cold storage units, reducing the need for physical presence.

## 3. Instant Alerts and Notifications

## IoT systems can be configured to send immediate alerts via SMS, email, or mobile notifications when environmental parameters go out of the acceptable range. This allows for quick action to prevent damage or spoilage.

## 4. Data Logging and Analytics

## All sensor readings are automatically recorded in databases or cloud storage. This historical data can be analyzed to detect patterns, inefficiencies, or recurring problems, helping in better decision-making and predictive maintenance.

From a deployment perspective, analytics dashboards can be created using platforms like Node-RED, Grafana, or Power BI to provide graphical interfaces for better interpretation of data. Alerts can also be linked to data logging—triggered when a rapid temperature spike is recorded or if the system detects prolonged deviation from ideal conditions. In mobile cold chain logistics, this logged data can serve as digital proof that the product was maintained under appropriate conditions throughout its journey, improving accountability and customer confidence.

In conclusion, integrating data logging and analytics transforms the cold storage monitoring system from a reactive tool into a powerful **intelligent management platform**.

# CHAPTER - 6

# CONCLUSION AND FUTURE SCOPE

## 6.1 Conclusion

The IoT-Based Cold Storage Monitoring System developed using the ESP32 microcontroller, DHT11 sensor, OLED display, and relay module stands as a practical and cost-effective solution to address the growing demand for real-time environmental monitoring in cold storage facilities. The project successfully demonstrated how embedded systems and IoT technologies can be integrated to automate and remotely manage critical storage parameters such as temperature and humidity. By providing both local visual feedback through an OLED display and remote monitoring via a web interface, the system ensures continuous visibility of cold storage conditions, empowering users to make timely decisions to prevent spoilage and maintain quality.

The automatic control mechanism using a relay adds a layer of intelligence to the system, enabling autonomous responses such as activating a fan or alert system when thresholds are crossed. This automation greatly reduces human dependency and allows for 24/7 surveillance of storage environments. Throughout the implementation and testing phase, the system proved to be stable, responsive, and accurate, fulfilling its primary objectives. Its modular nature allows for easy scalability, making it suitable not just for small-scale cold rooms, but also for larger commercial warehouses with minor adjustments and additions.

Moreover, the system addresses key industry challenges such as manual errors, energy inefficiency, and lack of timely alerts. It also demonstrates how modern IoT solutions can be made accessible to rural and small business owners, thanks to the use of affordable and easily available components. The success of this project showcases the potential of IoT in improving food security, medical supply preservation, and post-harvest agricultural management. Overall, this Cold Storage Monitoring System is a reliable, efficient, and user-friendly platform that aligns with the ongoing shift toward smart infrastructure in essential sectors. It lays the groundwork for further enhancements, including cloud data storage, multi-zone monitoring, mobile app integration, and advanced predictive analytics for proactive maintenance and control.

# REFERENCES

[1] A. Bahga and V. Madisetti, *Internet of Things: A Hands-On Approach*, Universities Press, 2014.

[2] S. Ramesh and V. Karthik, “Smart Agriculture Monitoring System Using IoT,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 8, no. 6, pp. 1–4, 2019.

[3] R. Patel and A. Sharma, “IoT Based Smart Cold Chain Monitoring System,” *International Journal of Scientific Research in Engineering and Management (IJSREM)*, vol. 4, no. 2, pp. 1–6, 2020.

[4] K. A. Naik, M. S. Palkar, and A. G. Ambavade, “Cold Storage Monitoring System Using IoT,” in *2021 International Conference on Communication information and Computing Technology (ICCICT)*, Mumbai, India, 2021, pp. 1–6. doi: 10.1109/ICCICT50803.2021.9509869

[5] P. Malviya, S. Singh, and N. Sinha, “Smart Cold Chain Management Using Internet of Things,” in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, 2020, pp. 543–548. doi: 10.1109/ICACCS48705.2020.9074296

[6] R. Kumar and M. P. Sarma, “Real-Time Temperature and Humidity Monitoring Using Arduino and Wi-Fi,” in *2018 3rd International Conference for Convergence in Technology (I2CT)*, Pune, India, 2018, pp. 1–4. doi: 10.1109/I2CT.2018.8529556

[7] T. Sharma and N. A. Gururaj, “IoT Based Temperature and Humidity Monitoring System for Cold Storage,” *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 9, no. 3, pp. 253–258, 2020.

[8] A. Kumar and A. K. Singh, “A Review on Cold Chain Monitoring Using Wireless Sensor Networks,” *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 9, no. 1, pp. 5259–5263, Oct. 2019.

# APPENDIX

#define BLYNK\_TEMPLATE\_ID "TMPL3iwW28ish"

#define BLYNK\_TEMPLATE\_NAME "my project"

#define BLYNK\_AUTH\_TOKEN "2C0tsJr\_LqbKjkM2w3cjtmFBMA\_zUnx8"

#include <WiFi.h>

#include <BlynkSimpleEsp32.h>

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <DHT.h>

// ==== Wi-Fi credentials ====

char ssid[] = "cold storage";

char pass[] = "123456789";

// ==== Pin Definitions ====

#define DHTPIN 4

#define DHTTYPE DHT11

#define GAS\_SENSOR\_PIN 34

#define LDR\_PIN 27

#define RELAY\_PIN 26

#define BUZZER\_PIN 25

// ==== Thresholds ====

#define TEMP\_THRESHOLD 50

#define GAS\_THRESHOLD 500

// ==== Objects ====

DHT dht(DHTPIN, DHTTYPE);

LiquidCrystal\_I2C lcd(0x27, 16, 2); // Use 0x27 or 0x3F depending on your LCD module

void setup() {

Serial.begin(115200);

// Initialize LCD

lcd.begin();

lcd.backlight();

// Start DHT sensor

dht.begin();

// Set pin modes

pinMode(RELAY\_PIN, OUTPUT);

pinMode(BUZZER\_PIN, OUTPUT);

pinMode(LDR\_PIN, INPUT);

digitalWrite(RELAY\_PIN, HIGH); // Relay OFF

digitalWrite(BUZZER\_PIN, LOW); // Buzzer OFF

// Connect to WiFi and Blynk

Blynk.begin(BLYNK\_AUTH\_TOKEN, ssid, pass);

// LCD Welcome Message

lcd.setCursor(0, 0);

lcd.print(" Cold Storage ");

lcd.setCursor(0, 1);

lcd.print(" Monitoring ");

delay(2000);

lcd.clear();

}

void loop() {

Blynk.run();

float temperature = dht.readTemperature();

float humidity = dht.readHumidity();

int gasValue = analogRead(GAS\_SENSOR\_PIN);

int ldrState = digitalRead(LDR\_PIN); // LOW = dark, HIGH = light

// Display on Serial Monitor

Serial.print("Temp: "); Serial.print(temperature);

Serial.print(" Hum: "); Serial.print(humidity);

Serial.print(" Gas: "); Serial.print(gasValue);

Serial.print(" LDR: "); Serial.println(ldrState == LOW ? "open" : "close");

// Update LCD

lcd.setCursor(0, 0);

lcd.print("T:"); lcd.print(temperature, 1); lcd.print("C ");

lcd.print("H:"); lcd.print(humidity, 0); lcd.print("%");

lcd.setCursor(0, 1);

lcd.print("G:"); lcd.print(gasValue);

lcd.print(" L:"); lcd.print(ldrState == LOW ? "open " : "close");

// Alert conditions

bool alert = (temperature > TEMP\_THRESHOLD) || (gasValue > GAS\_THRESHOLD) || (ldrState == LOW);

digitalWrite(RELAY\_PIN, alert ? LOW : HIGH); // Relay ON if alert

digitalWrite(BUZZER\_PIN, alert ? HIGH : LOW); // Buzzer ON if alert

// Send to Blynk

Blynk.virtualWrite(V0, temperature);

Blynk.virtualWrite(V1, humidity);

Blynk.virtualWrite(V2, gasValue);

Blynk.virtualWrite(V3, alert ? 1 : 0); // Relay status

Blynk.virtualWrite(V4, ldrState == LOW ? 1 : 0); // LDR status

delay(2000);

}